Aplicação de *A-Teams* ao Problema de Recobrimento

Humberto José Longo

Aplicação de *A-Teams* ao Problema de Recobrimento

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida pelo Sr. Humberto José Longo e aprovada pela Comissão Julgadora.

Campinas, 26 de outubro de 1995.

Prof. Marcus Vinicius S. Poggi de Aragão
Orientador

Dissertação apresentada ao Instituto de Matemática, Estatística e Ciência da Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aplicação de *A-Teams* ao Problema de Recobrimento[1]

Humberto José Longo[2]

Departamento de Ciência da Computação
IMECC — UNICAMP

Banca Examinadora:

- Celso Carneiro Ribeiro [3]
- Marcus Vinicius S. Poggi de Aragão (Orientador)[4]
- Cid Carvalho de Souza [4]
- João Meidanis (Suplente) [4]

[1] Dissertação apresentada ao Instituto de Matemática, Estatística e Ciência da Computação da UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

[2] O autor é Bacharel em Ciência da Computação pela Universidade Federal de Goiás.

[3] Professor do Departamento de Informática - PUC - RJ.

[4] Professor do Departamento de Ciência da Computação - IMECC - UNICAMP.

Aos meus Pais.

Agradecimentos

Ao final deste trabalho devo agradecer às muitas pessoas que colaboraram de alguma forma para a sua realização. Agradecer a todos, nominalmente, é impossível. Entretanto, algumas menções devem ser feitas:

Merece especial referência a assistência dada pelo Prof. Marcus Vinicius S. Poggi de Aragão, que pacientemente orientou e acompanhou o desenvolvimento deste projeto.

Agradeço também aos Profs. Celso Carneiro Ribeiro, Cid Carvalho de Souza e João Meidanis, pela prestativa participação na banca examinadora e valiosos comentários e sugestões.

Aos amigos Hélvio Pereira Peixoto, Elaine Gaspareto Haddad, Anderson Délcio Parreira e Victor Fernandes Cavalcante pelas inúmeras vezes em que os incomodei durante a realização deste projeto. A paciência e presteza demonstrada pelos mesmos nos momentos em que precisei de suas colaborações foram imprescindíveis para o bom andamento do trabalho.

Aos colegas do Departamento de Estatística e Informática (DEI) da Universidade Federal de Goiás (UFG), que superaram enormes dificuldades e pressões e mantiveram as atividades do departamento, sem prejuízos ao programa de qualificação de seus docentes.

A Universidade Federal de Goiás e à Capes pelo apoio financeiro recebido.

" Science is nothing but trained and organized commom sense ..."

T. H. Huxley, 1878

Resumo

Esta dissertação tem como tema central o Problema de Recobrimento de um Conjunto (*SCP - Set Covering Problem*). O objetivo principal é a proposta de uma nova abordagem para sua resolução, mais precisamente, este objetivo visa o desenvolvimento de um método heurístico, multi-algorítmico, baseado no paradigma de Times Assíncronos.

Um segundo objetivo desta dissertação, e de grande importância na fundamentação do método ora proposto, é um estudo das principais características estruturais do problema; de sua formulação como um problema de programação linear inteira 0-1 e dos principais métodos computacionais (heurísticos e exatos) atualmente disponíveis para sua resolução.

Times Assíncronos são organizações de *software* que visam a interação eficiente entre vários algoritmos, para a resolução de problemas adequados à abordagem multi-algorítmica. A arquitetura proposta utiliza métodos aproximados para a resolução do SCP e do dual da relaxação linear do mesmo. Esta abordagem primal-dual permite garantir que a melhor solução encontrada esteja a um certo percentual da solução ótima, ou mesmo, eventualmente, provar a otimalidade da solução. Segundo este enfoque, os principais componentes da arquitetura proposta são algoritmos gulosos e de consenso, procedimentos de busca tabu, métodos de otimização por subgradientes e geradores de planos de corte.

Os principais métodos exatos para a resolução do SCP são baseados em metodologias enumerativas. A maioria desses métodos combina ao esquema de enumeração diversas das técnicas heurísticas utilizadas na arquitetura aqui proposta. Contudo, esses métodos apresentam desempenho insatisfatório para algumas classes de instâncias, por não obterem boas soluções em um limite razoável de tempo.

A arquitetura proposta foi aplicada à instâncias dessas classes de difícil resolução. Os resultados obtidos mostraram que é possível alcançar, com um esforço computacional aceitável, resultados no mínimo comparáveis aos dos melhores algoritmos para o SCP.

Abstract

The development of an Asynchronous Team Method for heuristic resolution of the Set Covering Problem (SCP) is the main focus of this dissertation. Asynchronous Teams are software organizations that aim to efficient interaction among several algorithms for the resolution of problems that fit in a multi-algorithm approach.

Another goal of this work is an extensive study of the SCP which covers: the SCP structure; its formulation as a 0-1 ILP; and the description of the main heuristic and exact methods currently available for its resolution. This study is mostly required since we are concerned with the development of a multi-algorithm method.

The resulting software architecture makes use of approximate algorithms for the resolution of the SCP and its continuous relaxation dual. This primal-dual approach guarantees the best found solution to be at a certain percentage of the optimal solution and, eventually, proves the solution optimality. The main components of the proposed architecture are greedy and consensus algorithms, tabu search procedures, subgradient methods and cutting plane generators.

The main exact methods for the SCP resolution are based on enumerative methodologies. Most of these methods deploys many of the heuristic technics used in the proposed architecture to the enumeration scheme. However, these methods have a poor performance in some instance classes, because they do not obtain good solutions in a reasonable time limit.

The proposed architecture was applied to particularly hard instances. The obtained results show that it is possible to reach solutions, at an acceptable computational effort, that are at least comparable to the ones obtained by the best algorithms for the SCP.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Otimização Combinatória

Os problemas de otimização dividem-se, basicamente, em duas categorias: aqueles com variáveis contínuas e os com variáveis discretas. Tais problemas são de importância crescente devido ao grande número de problemas práticos que podem ser formulados e resolvidos como problemas de otimização.

Otimização Combinatória é um termo relativamente recente, surgido para unificar tópicos que abrangem Programação Inteira, Teoria dos Grafos e partes de Programação Dinâmica. O título Otimização Combinatória descreve as áreas de Programação Matemática relacionadas com a resolução de problemas de otimização que tem uma estrutura marcadamente discreta ou combinatorial. Um problema geral de Programação Matemática pode ser definido como: $min f(x)$, $x \in S \subseteq R^n$, onde f é chamada de função objetivo e S é o conjunto de soluções associadas ao problema. O objetivo da Programação Matemática é determinar se, para um certo problema, existe solução e, no caso afirmativo, encontrar uma ou todas as soluções ótimas.

Uma ramificação da Programação Matemática é a Programação Inteira, na qual $S \subseteq Z^n \subseteq R^n$ é o conjunto de soluções possíveis do problema. Sendo linear a função objetivo $f(x)$ do problema e S definido por um sistema de restrições lineares, tem-se um problema de Programação Linear Inteira. Os problemas combinatórios podem ser formulados, de modo mais ou menos complicado, como um problema de programação linear inteira (PLI). Essa formulação é particularmente interessante quando a solução de uma relaxação do problema, uma que considere um espaço definido por restrições lineares (um poliedro) S' onde $S' \supset S$, pode ser usada de forma eficiente na busca da solução inteira.

A Programação Linear Inteira é um modelo muito abrangente. Embora existam técnicas gerais que lidam com essa estrutura, muito trabalho tem sido desenvolvido no intuito de obter-se algoritmos especiais para resolver subclasses particulares desse modelo geral, como por exemplo o Problema de Recobrimento

de um Conjunto. Neste caso particular, contudo, depois de décadas de esforço dedicado à resolução do problema, ainda não é conhecido um algoritmo eficiente para resolver grandes instâncias do mesmo. Nesta dissertação será descrito um método multi-algorítmico, não necessariamente seqüencial, para tratar este caso particular.

1.2 Recobrimento de um Conjunto

O problema de recobrimento de um conjunto é equivalente à busca pelo menor número de subconjuntos, de um determinado conjunto, que unidos geram esse conjunto principal. De maneira precisa tem-se que, dado um conjunto $I = \{1, \ldots, m\}$ e uma família $F = \{I_1, \ldots, I_n\}$ de subconjuntos de I associada a $J = \{1, \ldots, n\}$, qualquer subconjunto $J^* \subseteq J$ define um recobrimento de I, se $\bigcup_{j \in J^*} I_j = I$ [FR61, Edm62]. Um recobrimento J^* é dito redundante se $J^* - \{j\}$ $(j \in J^*)$ ainda define um recobrimento. Um caso especial de recobrimento, dito particionamento de I, é obtido quando J^* satisfaz $I_j \bigcap I_k = \emptyset$ para $j, k \in J^*$ $(j \neq k)$.

Associando-se a cada conjunto j da família F um custo c_j, um recobrimento J^* terá um custo total de $\sum_{j \in J^*} c_j$. O interesse é encontrar um recobrimento de custo mínimo. Este problema de minimização é chamado de Problema de Recobrimento de um Conjunto (*SCP - Set Covering Problem*) [Law66]. Os custos c_j são sempre positivos e em geral distintos. Entretanto, uma classe de instâncias muito estudada é aquela em que todos os custos c_j são iguais a 1 (*unicost*). Nesta dissertação será tratado o SCP geral, mas devido a importância do SCP de custo unitário, eventualmente serão feitas referências específicas a esse caso especial.

O Problema de Recobrimento de um Conjunto é computacionalmente difícil de ser resolvido e claramente pertencente à classe dos problemas NP-difíceis. Karp [Kar72] mostrou que o problema de se encontrar uma cobertura dos vértices de um grafo G reduz-se trivialmente ao problema de se encontrar um recobrimento de um conjunto I, do seguinte modo: dado um grafo $G = (V, E)$, para $u, v \in V$, o elemento $(u, v) \in I$ se existir o arco $(u, v) \in E$ e $I_j \subseteq I$ for o conjunto de arcos incidentes ao vértice j.

Reforçando a dificuldade da resolução do Problema de Recobrimento de um Conjunto, Garey e Johnson [GJ79] observam que, mesmo se $|I_j| \leq 3$, $\forall\ I_j \subseteq I$, o problema continua pertencendo à classe NP-Completo. Só é possível garantir que seja resolvido em tempo polinomial se $|I_j| \leq 2$, $\forall\ I_j \subseteq I$. Os métodos propostos por Norman e Rabin [NR59] e Edmonds [Edm65], por exemplo, resolvem este caso especial de forma bastante eficiente.

1.3 Aplicações do *SCP*

O Problema de Recobrimento de um Conjunto é de importância fundamental em Otimização Combinatória. Sua importância vem de dois fatores [KS90]:

1. O seu modelo teórico. Em particular suas interconexões com outros ramos da matemática discreta, tais como hipergrafos, funções boolenas e satisfatibilidade.

2. O grande número de situações reais que podem ser modeladas com o problema.

Uma situação real pode ser formulada como um Problema de Recobrimento de um Conjunto definindo-se apropriadamente:

- um conjunto finito I que represente ações a serem executadas, decisões a serem tomadas ou alguma outra modelagem de uma situação real;

- uma família F de subconjuntos de I, onde cada subconjunto represente diferentes meios, recursos ou métodos para atingir, total ou parcialmente, o objetivo desejado; e

- um custo associado a cada membro de F.

A busca da melhor solução para a situação real reduz-se então a encontrar um conjunto de membros de F que seja de custo mínimo e recubra o conjunto I.

Tomando-se, por exemplo, o problema de recuperação de informações em n arquivos I_j [GN72], onde $c_j = |I_j|$, $j = 1, \ldots, n$, é o tamanho de cada arquivo. Cada unidade de informação i é armazenada em pelo menos um arquivo I_j ($i \in I_j$). Supondo-se existirem m requisições de informações, então um recobrimento ótimo fornece um subconjunto de arquivos, minimizando o volume total de informações que necessita ser pesquisado para garantir a recuperação das informações requisitadas.

Dentre as situações que podem ser modeladas com esse problema estão: definição de escalas para tripulações de linhas aéreas, roteamento de veículos, recuperação de informações, investimento de capital, projeto de circuitos, coloração de mapas, análise PERT/CPM e lógica simbólica, dentre outras. Uma extensa bibliografia a respeito dessas situações pode ser encontrada em [BP76, SM89, FK90, BJ92].

1.4 Organização da Dissertação

A modelagem do SCP como um problema de programação linear inteira, as principais propriedades decorrentes dessa modelagem, alguns problemas diretamente relacionados e aspectos do politopo associado ao SCP são apresentados

no capítulo 2. No capítulo 3 é feito um resumo dos principais métodos heurísticos aplicáveis ao SCP. O capítulo 4 descreve as principais características de alguns algoritmos exatos, enumerativos e baseados em planos de cortes, atualmente disponíveis para a resolução do SCP. No capítulo 5 é feita a descrição da arquitetura de Times Assíncronos, os quais constituem uma nova abordagem algorítmica para resolução de problemas combinatórios. Ainda neste capítulo são introduzidos modelos gerais aplicáveis ao SCP e é proposta uma arquitetura específica para resolução do SCP. No capítulo 6 são descritos os resultados computacionais da aplicação dos modelos propostos à várias classes de instâncias. Finalmente, o capítulo 7 apresenta algumas conclusões a respeito da aplicabilidade de Times Assíncronos ao SCP e propõe futuras extensões a este trabalho.

Capítulo 2

Estrutura e Propriedades do *SCP*

2.1 Formalização do *SCP*

Um problema geral de Programação Linear Inteira pode ser formulado como:

$$
\begin{array}{lllll}
 & \min(\max) & \sum_{j=1}^{n} c_j x_j & & \\
\textbf{(P)} & \text{s.a.} & \sum_{j=1}^{n} a_{ij} x_j & \left\{ \begin{array}{c} \leq \\ = \\ \geq \end{array} \right\} & b_i \quad i \in M = \{1, \ldots, m\} \\
 & & x_j & \geq & 0 \quad j \in N = \{1, \ldots, n\} \\
 & & x_j & \in & Z^n \quad j \in I \subset N
\end{array}
$$

Se $I = N$, ou seja, todas as variáveis são restritas a valores inteiros, o problema (P) é dito inteiro puro. Caso contrário, se $I \subset N$, (P) é dito inteiro misto.

O Problema de Recobrimento de um Conjunto pertence à subclasse dos problemas de Programação Linear Inteira 0-1. Para a formalização do SCP a variável $x_j e os dados a_{ij}$ e b_i, para $j = 1, \ldots, n$ e $i = 1, \ldots, m$ são definidas como:

$$x_j = \begin{cases} 1 & \textit{se o conjunto } j \textit{ está no recobrimento}, \\ 0 & \textit{caso contrário}; \end{cases}$$

$$a_{ij} = \begin{cases} 1 & \textit{se o elemento } i \textit{ pertence ao conjunto } j, \\ 0 & \textit{caso contrário}; \end{cases}$$

$$b_i = 1.$$

O Problema de Recobrimento de um Conjunto pode agora ser escrito como

segue:

$$\begin{array}{lrcll} \min & \sum_{j=1}^{n} c_j x_j & & & \\ \text{s.a.} & \sum_{j=1}^{n} a_{ij} x_j & \geq & 1 & i = 1, \ldots, m \\ & x_j & \in & \{0,1\} & j = 1, \ldots, n \end{array} \tag{2.1}$$

Uma solução x para esse problema é um vetor de tamanho n que satisfaça as restrições descritas acima. Uma tal solução x é dita não redundante se pelo menos uma dessas restrições deixa de ser satisfeita ao se fazer $x_j = 0$ para qualquer $j \in \{1, \ldots, n\}$.

Em vários tópicos dessa dissertação, será mais conveniente discutir o SCP em termos de conjuntos e recobrimentos do que desigualdades e variáveis. As definições a seguir fornecem o suporte teórico básico para a discussão do SCP em termos de conjuntos:

- $I = \{1, \ldots, m\}$: índices referentes às restrições;
- $J = \{1, \ldots, n\}$: índices referentes às variáveis;
- $I_j = \{i \in I \mid a_{ij} = 1\},\ j \in J$: restrições nas quais a variável x_j aparece; e
- $J_i = \{j \in J \mid a_{ij} = 1\},\ i \in I$: variáveis contidas na restrição i.

2.2 Problemas Relacionados

2.2.1 *SPP* e *SP*

Considerando-se o SCP como o modelo abstrato descrito na seção 1.2, existem dois outros problemas diretamente ligados a ele: o Problema de Particionamento de um Conjunto (*SPP - Set Partitioning Problem*) e o (*SP - Set Packing Problem*).

Usando-se a notação definida na seção 1.2, o SP reduz-se a se encontrar o maior conjunto J^* tal que $\bigcup_{j \in J^*} I_j \subseteq I$ e $\bigcap_{j \in J^*} I_j = \emptyset$, ou seja, o problema é unir o máximo de subconjuntos da família F sem que haja interseção entre eles. O SPP é um caso especial do SCP que é obtido quando um recobrimento J^* satisfaz $I_j \bigcap I_k = \emptyset$ para $j, k \in J^*$ $(j \neq k)$, ou seja, o problema é encontrar um recobrimento onde a interseção entre os subconjuntos seja vazia.

Analogamente ao SCP, esses dois últimos podem ser modelados como problemas de programação linear inteira. Observando-se que no SP o objetivo é maximizar o número de subconjuntos utilizados, sua formalização é a seguinte:

$$\begin{array}{lrcll} \max & \sum_{j \in J} c_j x_j & & & \\ \text{s.a.} & \sum_{j \in J_i} x_j & \leq & 1, & i \in I \\ & x_j & \in & \{0,1\}, & j \in J \end{array} \tag{2.2}$$

e a formalização do SPP:

$$\begin{array}{lrcll} \min & \sum\limits_{j \in J} c_j x_j & & & \\ \text{s.a.} & \sum\limits_{j \in J_i} x_j & = & 1, & i \in I \\ & x_j & \in & \{0,1\}, & j \in J \end{array} \tag{2.3}$$

As formalizações do SCP e do SP como problemas de programação linear inteira são bastante semelhantes e contêm uma importante propriedade. Sendo os dois problemas de custos unitários, então relaxando-se a condição binária nas variáveis em ambos, tem-se que a relaxação do SP é o dual da relaxação do SCP.

Uma propriedade comum aos três problemas (SCP, SPP e SP) é a solução ótima de cada um sempre ser um ponto extremo para a relaxação linear correspondente. Na seção 2.4 é descrita, para o SCP, essa propriedade e algumas outras correlatas.

Certas equivalências entre os três problemas são observadas com a adoção do modelo matemático descrito. O SPP pode ser convertido tanto para o SCP quanto para o SP. A conversão do SPP para SCP é descrita na próxima seção (2.2.2) e a conversão do SPP para o SP é análoga.

2.2.2 Conversão de *SPP* a *SCP*

Todo SPP com solução factível pode ser convertido em um SCP equivalente. No contexto, equivalência significa ambos terem a mesma solução ótima. O procedimento de conversão e sua demonstração foram extraídos de Lemke, Salkim e Spielberg [LSS71], Garfinkel e Nemhauser [GN72] e Taha [Tah75].

O procedimento de conversão dos problemas consiste na simples troca do vetor de custos c e, obviamente, na mudança das equações para inequações. Para tanto, sejam $t_j = \sum_{i=1}^{m} a_{ij},\ j = 1, \ldots, n$, e L um número tal que $L > \sum_{j=1}^{n} c_j$. O SCP obtido de um SPP com a mudança do vetor de custos c para $c'_j = c_j + Lt_j,\ j = 1, \ldots, n$, possui o mesmo conjunto de soluções ótimas do problema original. A demonstração desse procedimento mostra que qualquer solução factível para o SCP, e infactível para o SPP, tem valor maior do que a melhor solução factível para o SPP.

Sejam S_c e S_p os conjuntos de soluções para o SCP e SPP, respectivamente, observe que $S_c \supset S_p$ (se $S_c = S_p$ a demonstração é trivial). Tomando-se quaisquer $x^p \in S_p$ e $x^c \in S_c - S_p$, então:

$$\sum_{j \in J} c'_j x^c_j = \sum_{j \in J} c_j x^c_j + L \sum_{j \in J} t_j x^c_j$$

Por definição:

$$\begin{array}{rcll} \sum\limits_{j \in J_i} x^c_j & \geq & 1 & i \in I \\ & = & 1 + \Delta_i & \end{array}$$

Logo:

$$\sum_{j\in J} c'_j x^c_j \geq L \sum_{i\in I}(\sum_{j\in J_i} x^c_j) = L \sum_{i\in I}(1+\Delta_i) \geq L(m+1)$$

Por outro lado:

$$\begin{aligned}\sum_{j\in J} c'_j x^p_j &= \sum_{j\in J} c_j x^p_j + L \sum_{i\in I}(\sum_{j\in J_i} x^p_j) \\ &= \sum_{j\in J} c_j x^p_j + Lm \leq \sum_{j\in J} c_j + Lm < L + Lm = L(m+1)\end{aligned}$$

Isto mostra que $c'x^c > c'x^p$, significando que uma solução factível apenas para o SCP não pode ser ótima para o SPP.

Esse procedimento de conversão admite que o SPP tem solução factível, o que nem sempre é correto, ao contrário do SCP. Como pode ser difícil determinar de antemão se um SPP é infactível, é necessário verificar, após a resolução do SCP equivalente, se a solução obtida é factível para o SPP.

O procedimento anterior permite a resolução de instâncias do SPP por meio de um algoritmo especialmente construído para o SCP, contudo, pode acarretar em sérios erros de arredondamento na implementação computacional. Esse problema é particularmente suscetível de ocorrer na soma de custos grandes, relativamente aos custos individuais, ou seja, quando o vetor de custos é composto por valores pequenos e próximos uns dos outros [SM89].

2.2.3 Modelagens de Problemas em Grafos como Problemas em Conjuntos

Muitos problemas definidos em grafos podem ser modelados como problemas relacionados a conjuntos, ou seja, como SCP, SPP ou SP [SM89]. Nesta seção é descrita a equivalência entre alguns desses problemas e o SCP (ou SP). Para tanto, considera-se que um grafo $G = (V, E)$ é composto de uma coleção V de vértices e E de arestas que os unem:

- **Recobrimento de vértices :** o problema de recobrimento dos vértices de um grafo é equivalente a se encontrar o menor subconjunto de arcos, tal que cada vértice do grafo seja extremo de pelo menos um arco desse subconjunto. Definindo-se $x_j = 1$, $j \in E$, se o arco j pertence ao recobrimento, e $x_j = 0$ caso contrário; $a_{ij} = 1$, $i \in V$ e $j \in E$, se o vértice i é extremo do arco j, e $a_{ij} = 0$ caso contrário, então claramente esse problema é equivalente a um SCP de custos unitários.

- **Emparelhamento de arestas :** o problema de emparelhamento em grafos é o de se encontrar o maior subconjunto de arcos (arestas), tal que quaisquer dois arcos desse subconjunto não tenham extremos em comum. Seja $x_j = 1$, $j \in E$, se o arco j pertence ao emparelhamento, e $x_j = 0$ caso contrário;

$a_{ij} = 1$, $i \in V$ e $j \in E$, se o vértice i é extremo do arco j, e $a_{ij} = 0$ caso contrário. Cada vértice deve ser extremo de no máximo um arco do emparelhamento, o que é garantido pelas restrições $\sum_{j \in E} a_{ij} \leq 1$, $i \in V$. Como o objetivo é maximizar o número de arcos no emparelhamento, então esse problema é equivalente ao SP e possui algoritmo de complexidade polinomial (Edmonds [Edm65]).

- **Corte de cardinalidade mínima:** supondo-se conhecidos todos os caminhos (seqüências de vértices distintos) entre dois vértices s e t, o problema é encontrar o menor subconjunto de arcos tal que, se removidos do grafo, todos os caminhos tornem-se desconexos. Redefinindo-se V como o conjunto de caminhos entre s e t, então $a_{ij} = 1$ se o arco j pertence ao caminho i, e 0 caso contrário. Definindo-se $x_j = 1$, $j \in E$, se o arco j for removido, e $x_j = 0$ caso contrário, então o problema torna-se equivalente ao SCP, onde as restrições $\sum_{j \in E} a_{ij} \geq 1$, $i \in V$ garantem que pelo menos um arco é excluído de cada caminho.

2.3 Propriedades e Regras de Redução

Esta seção lista algumas das propriedades fundamentais do SCP, bem como regras para testar a factibilidade ou reduzir o tamanho do problema. Essas regras basicamente identificam variáveis que podem ser fixadas em 0 ou 1 sem perda da otimalidade. Estes testes são normalmente utilizados em uma fase de pré-processamento nos algoritmos propostos para a resolução do SCP. O conteúdo dessa seção é baseado nos trabalhos de Lemke, Salkim e Spielberg [LSS71] (propriedades 1 a 6) e Fisher e Kedia [FK90] (propriedade 7).

1. **Custos positivos.** Pode-se assumir que $c_j > 0$, $\forall\, j \in J$. Se $c_j \leq 0$, então pode-se fazer $x_j = 1$, e excluir a coluna j e, conseqüentemente, toda linha $i \in I$, tal que $a_{ij} = 1$, reduzindo-se assim o tamanho do problema.

2. **Valores de x_j.** É suficiente considerar que $x_j \in \{0, 1\}$, $\forall\, j \in J$. Se x é uma solução factível com $x_j > 1$ para algum j, fazendo-se $x_j = 1$ a solução continua factível, porém com custo menor.

3. **Factibilidade.** O problema é infactível se $|J_\alpha| = 0$ para algum $\alpha \in I$. Claramente a restrição $\sum_{j \in J_\alpha} x_j \geq 1$ não pode ser satisfeita.

4. **Variáveis pré-fixadas.** Se $|J_\alpha| = 1$ para algum $\alpha \in I$, pode-se fazer $x_j = 1$ para o único $j \in J_\alpha$. Como conseqüência, pode-se excluir toda linha $\beta \in I_j$, pois todas as restrições associadas a tais linhas já estarão satisfeitas pela variável j.

5. **Dominância de linha.** Se para $\alpha, \beta \in I$ tem-se que $J_\beta \subseteq J_\alpha$, então pode-se fixar $x_j = 0$ para todo $j \in (J_\alpha \setminus J_\beta)$ e tal que $j \notin J_\delta$ ($\delta \in I$ e $\delta \neq \alpha$),

e excluir a linha α pois a restrição associada a ela será satisfeita toda vez que o for a restrição associada à linha β.

6. **Dominância de coluna.** Se para algum $C \subset J$ e $\alpha \in J$ tem-se que $I_\alpha \subseteq \bigcup_{j \in C} I_j$ e $\sum_{j \in C} c_j < c_\alpha$, então a coluna α pode ser excluída, pois o custo de incluí-la no recobrimento seria maior que o de inclusão de todas as colunas $j \in C$.

7. **Coluna de custo dominado.** Para todo $i \in I$, seja $d_i = (\min c_j \mid j \in J_i)$. Se para $\alpha \in J$ tem-se que $c_\alpha > \sum_{i \in I_\alpha} d_i$, então a coluna α pode ser excluída, pois as linhas por ela recobertas podem sê-las com menor custo por outras colunas.

2.4 Propriedades da Relaxação Contínua

Nesta seção são examinadas algumas das propriedades do seguinte problema de programação linear:

$$\begin{array}{lrcl} \min & \sum\limits_{j \in J} c_j x_j & & \\ \text{s.a.} & \sum\limits_{j \in J_i} x_j & \geq & 1, \quad i \in I \\ & x_j & \geq & 0, \quad j \in J \end{array} \tag{2.4}$$

e do problema dual associado a ele:

$$\begin{array}{lrcl} \max & \sum\limits_{i \in I} u_i & & \\ \text{s.a.} & \sum\limits_{i \in I_j} u_i & \leq & c_j, \quad j \in J \\ & u_i & \geq & 0, \quad i \in I \end{array} \tag{2.5}$$

O problema 2.4 é o mesmo 2.1, porém sem a restrição de condição binária nas variáveis. Portanto as propriedades a serem examinadas relacionam-se diretamente ao SCP. As referências para essas propriedades são Lemke, Salkim e Spielberg [LSS71], Glover [Glo71] e Bellmore e Ratliff [BR71].

1. **Uma solução ótima viável x, para o problema 2.4, satisfaz $x_j \leq 1$, $j \in J$.**

   Sejam $J' = \{j \in J \mid x_j > 1\}$ e $I'_j = \{i \in I \mid i \in I_j \ e \ x_j > 1, \ j \in J'\}$, então $\sum_{j \in J_i} x_j > 1$, $i \in I'_j$. Fazendo-se $x_j = 1$, $j \in J'$, tem-se que $\sum_{j \in J_i} x_j \geq 1$, $i \in I'_j$, ou seja, a solução continua factível. Por outro lado, como $c_j > 0$, $j \in J$, segue que se $x_j > 1$, $j \in J$, a solução não pode ser ótima.

2. **O problema 2.4 tem o mesmo conjunto de soluções ótimas que o problema obtido de 2.4 com a adição das restrições $x_j \leq 1$, $j \in J$.**

Conseqüência da propriedade 1.

3. **Se x' é solução factível para 2.4, então a solução x obtida com o arredondamento para cima de x' ($x_j = 1$ se $x'_j > 0$, $j \in J$) é factível para 2.1.**

   Como $x_j \geq x'_j$, $j \in J$, então $\sum_{j \in J_i} x_j \geq \sum_{j \in J_i} x'_j \geq 1$, $i \in I$, ou seja, x é factível para 2.1 pois satisfaz todas as restrições e $x_j \in \{0, 1\}$, $j \in J$.

4. **O problema 2.1 é factível se, e somente se, o problema 2.4 é factível.**

   Se x é solução factível para 2.1, claramente, tal solução também é factível para 2.4 ($x_j \geq 0$, $j \in J$ e $\sum_{j=1}^{n} a_{ij} x_j \geq 1$, $i \in I$). Por outro lado, se o problema 2.4 é factível, então, pela propriedade 3, o problema 2.1 também é factível.

5. **Todo ponto extremo x do conjunto de soluções de 2.4 satisfaz $0 \leq x_j \leq 1$, $j \in J$.**

   Tem-se da propriedade 1 que uma solução ótima x para 2.4 satisfaz $0 \leq x_j \leq 1$, $j \in J$, e da teoria de programação linear tem-se que tal solução é um ponto extremo. Por outro lado, como a solução ótima de 2.4 pode corresponder a qualquer um dos pontos extremos, dependendo do gradiente da função objetivo, implica que a propriedade é necessariamente verdadeira.

6. **Se x' é um ponto extremo (não integral) de 2.4, então o seu arredondamento x ($x_j = 1$ se $x'_j > 0$, $j \in J$) é uma solução redundante de 2.1.**

   Seja $J' = \{j \in J \mid 0 < x'_j < 1\}$ o conjunto de variáveis da solução x' com valores não inteiros, e I' o conjunto de restrições estritamente satisfeitas por tais variáveis (pelo menos duas variáveis $\alpha, \beta \in J'$ são necessárias para satisfazer cada uma dessas restrições). Claramente $|J'| \geq |I'|$. Portanto, ao se fazer $x_j = 1$ para apenas $|I'|$ dos $j \in J'$, mantém-se a factibilidade, enquanto na solução arredondada x é feito $x_j = 1$, $\forall\, j \in J'$.

7. **Se x é uma solução factível não redundante para 2.1, então x é um ponto extremo para 2.4.**

   Seja $J' = \{j \in J \mid x_j = 1\}$. Se x é não redundante, então $|J'| \leq |I|$. Como cada variável $j \in J'$ está em pelo menos uma restrição na igualdade, então as colunas especificadas por J' são linearmente independentes, o que equivale a dizer que x é ponto extremo.

8. **Toda solução ótima de 2.1 corresponde a um ponto extremo do conjunto de soluções factíveis de 2.4.**

   Conseqüência das propriedades 6 e 7.

9. **Se x' é solução factível para 2.4 e $H = \{j \in J \mid x'_j = 1$ e $\sum_{j \in J_i} x'_j > 1, \forall\ i \in I_j\} \neq \emptyset$, então para cada $j \in H$ existe uma solução x'' factível para 2.4 tal que $x''_j < x'_j$.**

   Seja $S_i = \{\sum_{j \in J_i} x'_j - 1\}$, $i \in I$ e $\theta_j = \min\{S_i \mid i \in I_j\}$. Tomando-se $j \in H$ e fazendo-se $x''_j = x'_j - min\{\theta_j, x'_j\}$, a factibilidade da solução é mantida.

10. **Se x' é uma solução factível para 2.4 e x é o seu arredondamento ($x_j = 1$ se $x'_j > 0$, $j \in J$), então a solução obtida atribuindo-se $x_j = 0$, para um j tal que $0 < x'_j < 1$, é factível para 2.1.**

    Para qualquer j tal que $0 < x'_j < 1$, tem-se que $\sum_{j \in J_i} x_j > 1$, $\forall\ i \in I_j$. Logo, pela propriedade 9, a variável x_j pode ser fixada em 0.

11. **Seja x uma solução não redundante para 2.1, $J' = \{j \in J \mid x_j = 1\}$ e B a matriz correspondente à base associada a essa solução, com a adição de $m - |J'|$ variáveis de folga $s_i = 1 - \sum_{j \in J_i} x_j$, $i \in I$, através das regras:**

    - **selecione s_i se $\sum_{j \in J'} a_{ij} \geq 2$;**
    - **para cada $j \in J'$ tal que $|I_j| > 1$, arbitrariamente selecione s_i correspondendo a qualquer um dos $|I_j| - 1$ elementos de I_j.**

    Existe uma matriz $\hat{B}$, derivada de B através de permutações apropriadas de linhas, tal que $\hat{B}^{-1} = \hat{B}$.

    O próprio mecanismo de construção da matriz B garante que esta sempre pode ter as linhas permutadas para que se obtenha a matriz $\hat{B} = \begin{bmatrix} I_1 & 0 \\ D & -I_2 \end{bmatrix}$, onde I_1 é de ordem $|J'|$ e I_2 de ordem $m - |J'|$. Logo, $\hat{B}.\hat{B} = \begin{bmatrix} I_1 & 0 \\ 0 & I_2 \end{bmatrix}$. Claramente a existência de $\hat{B}^{-1}$ implica a existência de B^{-1}. Diz-se que a matriz B possui a propriedade de involução.

12. **O problema de programação linear 2.5, dual ao problema 2.4, apresenta soluções de base factíveis.**

    Conseqüência direta da própria estrutura do problema 2.5.

As propriedades anteriores mostram que se o problema 2.1 tem uma solução factível que não é ponto extremo para 2.4, então uma solução melhor (em termos de valor da função objetivo) pode ser obtida transformando-se a solução original em ponto extremo para 2.4. Isto enfatiza a importância do problema 2.4 no desenvolvimento de algoritmos enumerativos e baseados em planos de cortes para o problema 2.1, já que a solução ótima desse último precisa ser ponto extremo para 2.4 [Tah75] (esta propriedade se estende a todos os problemas de programação linear inteira em variáveis 0-1).

2.5 Aspectos do Poliedro do *SCP*

O SCP foi extensivamente estudado nas últimas 3 décadas, contudo, esse estudo foi mais direcionado aos aspectos algorítmicos do problema. Somente mais recentemente começou a ser estudada a estrutura facial do problema, i.e. a descrição linear da envoltória convexa das soluções factíveis. Logo, para o SCP, como para a maioria dos problemas combinatórios, ainda não é conhecido um método sistemático para gerar todas as desigualdades lineares que definam a envoltória convexa das soluções para o problema. A seguir são relacionados os principais trabalhos nessa área. A terminologia e o conhecimento de base podem ser encontrados, por exemplo, em [Tah75, NW88].

O estudo das propriedades estruturais do poliedro do SCP recebeu pouca atenção na literatura, quando comparado a outros problemas combinatórios. Os primeiros resultados nessa área foram apresentados por Fulkerson [Ful71], Nemhauser e Trotter [NT74] e Padberg [Pad79]. Mais recentemente, a envoltória convexa das soluções factíveis do SCP foi parcialmente descrita através de resultados téoricos apresentados por Balas e Ng [BN89a, BN89b], Cornuéjols e Sassano [CS89], Sassano [Sas89] e Nobili e Sassano [NS92].

Em [BN89a] e [BN89b], Balas e Ng caracterizaram facetas, da envoltória convexa do SCP, definidas por desigualdades da forma $\sum_{j=1}^{n} \alpha_j x_j \geq 2$ ($\alpha_j \in \{0,1,2\}$, $j = 1,\ldots,n$). No segundo trabalho é mostrado que tais facetas podem ser obtidas através de "*lifting*" de certas desigualdades contendo apenas três coeficientes não nulos. As facetas definidas por desigualdades da forma $\sum_{j=1}^{n} \alpha_j x_j \geq \beta$ ($\alpha_j \in \{0,1\}$, $j = 1,\ldots,n$, e $\beta \in Z^+$), foram caracterizadas por Cornuéjols e Sassano [CS89, Sas89].

Sassano [Sas89] definiu uma condição suficiente, mas não necessária, para que uma desigualdade da forma citada anteriormente defina uma faceta. Este resultado foi estendido por Cornuéjols e Sassano [CS89], que definiram uma condição suficiente e necessária para que tais desigualdades sejam facetas, além de investigarem a que classe de instâncias do SCP elas definiriam completamente a envoltória convexa.

Nobili e Sassano [NS89] caracterizaram duas classes de facetas não booleanas. Os autores generalizaram ainda o conceito de "*web*", definido por Laurent [Lau89] para sistemas independentes, e descreveram uma classe de facetas produzidas por tais estruturas. Em [NS92] Nobili e Sassano propuzeram um algoritmo de separação para as facetas caracterizadas em [NS89].

A aplicação computacional das classes de facetas relacionadas anteriormente, ou seja, o seu uso como planos de cortes em algoritmos para resolução do SCP, foi pouco testada. Dos diversos trabalhos téoricos na área de descrição da estrutura facial do SCP, apenas um faz alguma referência a aplicações práticas, procurando mostrar que a abordagem poliédrica para a resolução do SCP pode ser muito eficaz. Sassano [Sas89] resolveu, embora manualmente, dois dos proble-

mas propostos por Fulkerson, Nemhauser e Trotter [FNT74], utilizando facetas, da classe descrita em seu trabalho, como planos de cortes.

Na abordagem proposta no capitulo 5 desta dissertação, para a resolução do SCP, são utilizadas desigualdades da forma $\sum_{j=1}^{n} \alpha_j x_j \geq 2$ ($\alpha_j \in \{0,1,2\}$) como planos de cortes. Um procedimento simples para gerar tais desigualdades é descrito em [BN89a]. Definindo-se $J_t(\alpha) = \{j \in J \mid \alpha_j = t\}$, $t = 0,1,2$ e $S \subset I$, obtém-se uma nova desigualdade válida para o SCP (da forma $\alpha x \geq 2$) fazendo-se :

$$\alpha_j = \begin{cases} 0 & se\ a_{ij} = 0\ \forall\ i \in S; \\ 2 & se\ a_{ij} = 1\ \forall\ i \in S;\ e \\ 1 & caso\ contrário. \end{cases}$$

Observe que se $|S| = 1$, ou seja $S = \{i\}$, a nova desigualdade corresponde à $i-$ésima restrição multiplicada por 2. Balas e Ng estabeleceram condições necessárias e suficientes para tais desigualdades definirem facetas.

Capítulo 3

Resolução Heurística do *SCP*

A importância dos métodos heurísticos para o SCP deve-se a dois fatores principais:

- muitas instâncias do SCP são extremamente difíceis de serem resolvidas até a otimalidade;
- A maioria dos métodos exatos utiliza métodos heurísticos para obtenção de boas soluções iniciais ou de limites superiores e inferiores no valor da solução ótima.

Existem na literatura diversos métodos heurísticos para resolução do SCP. Alguns dos principais estão descritos em [Rot69, Joh74, Chv79, Bak81, FW82, Ho82, Hoc82, VW84, VW88, FR89, KS90, EDM92]. Nas seções seguintes são descritos alguns desses métodos, os quais são partes integrantes dos algoritmos exatos descritos no capítulo 4.

Além de algoritmos exatos para o SCP, recentemente foram propostos diversos algoritmos heurísticos que basicamente são composições dos métodos descritos nas referências listadas anteriormente. Por exemplo, Beasley [Bea90] propôs um algoritmo heurístico constituído de vários desses métodos.

A aplicação de meta-heurísticas ao SCP também tem sido bastante investigada em diversos trabalhos recentes. Feo e Resende [FR95] descreveram a construção de um GRASP (*Greedy Randomized Adaptive Search Procedure*) voltado para o SCP. Beasley e Chu [BC94] e Sen [Sen93] propuseram algoritmos genéticos aplicados ao SCP. Jacobs e Brusco [JB93] e Sen [Sen93] desenvolveram heurísticas baseadas em *Simulated Annealing* para o SCP. Na seção 3.1.2 é descrita uma aplicação simples de busca tabu ao SCP.

3.1 Limites Superiores

3.1.1 Heurísticas de Construção

Existe na literatura um grande número de heurísticas de construção para o SCP. Essas heurísticas são essencialmente gulosas. O procedimento geral seleciona repetidamente, com base em algum critério, uma variável j que receberá valor 1. A parada se dá quando uma solução factível é obtida.

Considerando-se que o critério a ser usado para a escolha de uma variável é uma certa função f do coeficiente de x_j e da cardinalidade do conjunto I_j de linhas recobertas pela variável x_j, tais heurísticas têm a seguinte forma geral:

- **Heurística gulosa primal**

  1. $N = \{1, \ldots, n\}$;
     $x_j = 0$ e $I_j(x) = I_j,\ j \in N$;
  2. Se $I_j(x) = \emptyset$ para todo $j \in N$ então PARE (o vetor x é uma solução factível);
     caso contrário, encontre $k \in N$ que maximize a função $f(c_k, I_k(x))$;
  3. Faça $x_k = 1,\ N = N - \{k\}$ e $I_j(x) = I_j(x) - I_k(x),\ j \in N$;
     Retorne ao passo 2.

O uso de uma função diferente para a escolha de uma variável corresponde a uma heurística diferente. Chvátal [Chv79] propôs a função $f(c_j, I_j(x)) = c_j/|I_j(x)|$, escolhendo, em outras palavras, a variável que recobre o maior número de restrições, dentre as não recobertas, por unidade de custo. No caso em que todos os custos são unitários, ou seja, $c_j = 1,\ \forall\ j \in J$, a heurística proposta por Chvátal reduz-se às propostas por Johnson [Joh74] e Lovász [Lov75].

Essa mesma função e quatro outras ($c_j, c_j/\log_2|I_j(x)|, c_j/(|I_j(x)|\log_2|I_j(x)|)$ e $c_j/(|I_j(x)|\ln|I_j(x)|)$) foram analisadas por Balas e Ho [BH80]. Duas outras funções ($c_j/(|I_j(x)|)^2$ e $\sqrt{c_j}/(|I_j(x)|)^2$) foram propostas por Vasko e Wilson [VW84]. Uma variação, também proposta por Vasko e Wilson, é o uso intercalado de várias funções de escolha das variáveis na construção de uma mesma solução. Na heurística proposta por Roth [Rot69], a função de escolha das variáveis foi eliminada, sendo a escolha feita de modo aleatório.

Desse conjunto de heurísticas citado anteriormente, uma possível subclasse, é aquela formada por heurísticas que limitam as variáveis passíveis de serem escolhidas em cada iteração a um subconjunto do total de variáveis. O uso de diferentes critérios para a escolha de tais subconjuntos gera essa subclasse de heurísticas.

Balas e Ho [BH80] usaram uma heurística dessa subclasse onde cada subconjunto era formado pelas variáveis que compunham cada restrição do problema. O critério de escolha de um subconjunto foi o número de elementos em cada um,

sendo escolhido sempre o de menor cardinalidade. Outra modalidade de escolha de subconjuntos de variáveis foi proposta por Feo e Resende [FR89], onde a cada passo de escolha de uma variável, esta era escolhida dentre aquelas que recobriam um número mínimo de restrições.

Alguma inovação foi introduzida nessa classe de heurísticas por Fisher e Kedia [FK90], com a utilização de uma solução do problema dual da relaxação contínua do SCP (problema 2.5) para determinar o impacto das restrições do problema na escolha de variáveis. A Heurística proposta por eles segue o mesmo padrão da proposta por Balas e Ho, ou seja, em cada iteração uma variável é escolhida de um subconjunto (variáveis em uma restrição) do conjunto de variáveis. O subconjunto a ser escolhido é aquele que maximiza $u_i|J_i|$, onde u_i é a variável dual associada a restrição i e J_i é o conjunto de variáveis da restrição i. A regra para escolha de uma variável dentro de um subconjunto é similar à usada por Chvátal [Chv79], Lovász [Lov75] e Johnson [Joh74], exceto que c_j é trocado por $c_j - \sum_{i \in I_j} u_i$, onde I_j é o conjunto de restrições recobertas pela variável j.

A análise de desempenho de uma heurística para o SCP, pertencente à classe geral citada anteriormente, foi feita inicialmente por Chvátal [Chv79]. Chvátal mostrou que $Z_{HEU}/Z^* \leq \sum_{j=1}^{d} 1/j$ é o melhor limite possível, onde $d = \max_{j \in J} |I_j|$ e Z^* e Z_{HEU} são os valores da solução ótima e da encontrada pela heurística, respectivamente. Posteriormente Ho [Ho82] mostrou que o desempenho, no pior caso, de qualquer heurística dessa classe geral é dominado por aquele mostrado por Chvátal [Chv79]. Outras análises nesta área podem ser encontradas em [Joh74, Lov75, GJ79, BH80, Fis80, Wol80].

3.1.2 Heurísticas de Melhoria

As heurísticas de melhoria buscam o aprimoramento de uma solução factível, para um determinado problema, através de uma seqüência de trocas de partes (ou elementos) da solução, em uma busca local. Em outras palavras, busca-se, em iterações sucessivas, a melhor solução contida na vizinhança da solução corrente.

Denotando-se por $V_k(x)$ o conjunto de soluções factíveis vizinhas à solução x (diferem de x em não mais que k componentes), então x é um ótimo local se não existe em $V_k(x)$ uma solução melhor. Baseada nesses conceitos, uma heurística de busca local pode ser descrita como:

- **Heurística de busca local**

  1. Encontre uma solução inicial x;
  2. Se não existe $x' \in V_k(x)$, tal que x' é melhor que x, PARE x é ótimo local.
  3. Faça $x = x'$;
  4. Retorne ao passo 2;

A eficiência de uma heurística de busca local depende basicamente da escolha da vizinhança a ser considerada, da busca dentro dessa vizinhança e da solução inicial.

No caso do SCP, boas soluções iniciais podem ser obtidas conforme descrito na seção 3.1.1. A definição de vizinhança foi feita por Feo e Resende [FR95] em termos de uma *k,p*-troca, ou seja, dado p, k $(p < k)$ e um recobrimento x, então uma *k,p*-troca corresponde a exclusão de k variáveis que pertençam ao recobrimento e sua substituição por p outras não pertencentes ao mesmo. A especialização da busca local para o SCP de custos unitários proposta por Feo e Resende é:

- **Heurística *k,p-search***

  1. k, p são inteiros positivos satisfazendo $p < k < n - p$;
  2. x é uma solução factível para o SCP;
  3. $J' = \{j \in J \mid x_j = 1\}$;
  4. Para cada $T_1 \subset J'$ $(|T_1| = k)$:
  5. Verifique a existência de $T_2 \subset (J \setminus T_1)$ $(|T_2| = p)$;
  6. Se $(J' - T_1) \cup T_2$ configura uma solução factível, faça:
  7. $J' = (J' - T_1) \cup T_2$.

Essa heurística é uma generalização da proposta por Roth [Rot69], onde é adotado $p = k - 1$. Roth observou também que, quando os custos não são unitários, há sentido em se considerar $p \geq k$, pois mesmo assim um recobrimento de menor custo pode vir a ser obtido. Nesse caso a linha 5, na descrição anterior, deve ser alterada para:

5. Verifique a existência de $T_2 \subset (J \setminus T_1)$ $(|T_1| = p)$ com $\sum_{j \in T_2} c_j < \sum_{j \in T_1} c_j$;

Vasko e Wilson [VW84] e Vasko e Wolf [VW88] utilizaram heurísticas de melhorias que basicamente eliminavam colunas redundantes no recobrimento, o que é equivalente à heurística descrita anteriormente fixando-se $k = 1$ e $p = 0$.

A heurística *k,p-search* pode ser utilizada tanto isoladamente quanto integrada a outros métodos. Uma possibilidade é a sua integração na heurística de perturbação e melhoria descrita a seguir. Nessa heurística o processo de aprimoramento de uma solução é feito através de alterações sucessivas (perturbações) na mesma. O esquema de perturbação pode ser constituído de duas etapas. A primeira é a transformação da solução em outra redundante. A segunda é a obtenção de uma nova solução não redundante, a partir do descarte dos elementos excedentes.

Considerando-se $Z(x)$ como o valor objetivo da solução x, então esta heurística pode ser mais formalmente descrita como:

- **Heurística de perturbação de soluções**

  1. x é solução factível de valor objetivo $Z(x)$.
  2. Faça $x' = x$ e $Z(x') = Z(x)$.
  3. Repita k vezes:
  4. Seja $J' = \{j \in J \mid x'_j = 1\}$;
  5. Escolha $J'' \subset (J \setminus J')$ tal que $|J''| = \delta$;
  6. Faça $x'_j = 1 \; \forall \; j \in J''$;
  7. Aplique *k,p-search* a x';
  8. Se $Z(x') < Z(x)$ faça $x = x'$ e $Z(x) = Z(x')$.

Outra heurística de melhoria foi proposta por Baker [Bak81]. Nesta, através da composição de duas soluções factíveis, busca-se a formação de uma nova solução de menor custo. Nas duas soluções iniciais, a heurística identifica subconjuntos de colunas que recobrem subconjuntos disjuntos de linhas. Após todas as colunas terem sido fixadas a um dos subconjuntos, os custos destes são recalculados e os de menor custo são usados para se compor a nova solução.

Uma heurística utilizada na implementação computacional descrita no capítulo 6, também utiliza o conceito de composição de soluções factíveis. No caso, a composição consiste na determinação da interseção (variável a variável) entre um certo número de soluções. A solução parcial assim obtida é completada , se necessário, segundo algum esquema guloso, de modo a corresponder a um recobrimento.

Esse esquema geral permite diversas variações, dependendo do número de soluções e do critério de consenso utilizado. Adotando-se, por exemplo, o critério de igualdade (em 1) entre k diferentes soluções, então a formalização pode ser:

- **Heurística de consenso**

  1. $x^1, \ldots, x^k$ $(k \geq 2)$ são soluções factíveis;
  2. x^0 é a nova solução a ser obtida;
  3. Faça $x_j^0 = \begin{cases} 1 \; se \; x_j^1 = \ldots = x_j^k = 1 \\ 0 \; caso \; contrário; \end{cases}$, $j \in J$;
  4. Se x^0 não corresponde a um recobrimento, então utilize uma heurística gulosa primal (seção 3.1.1) para completá-lo.

A busca tabu é um procedimento adaptativo usado na resolução de problemas de otimização combinatória. O procedimento explora o espaço de soluções, para um determinado problema, a partir de uma solução inicial, obtendo uma seqüência de soluções através de modificações sucessivas nas mesmas. Essa seqüência de soluções é obtida observando-se um critério guloso na modificação da

solução corrente, ou seja, buscando-se sempre a melhor solução possível. Contudo, são consideradas apenas as soluções que podem ser alcançadas dentro de uma vizinhança restrita. Neste processo pode ocorrer, entre outros problemas, a ciclagem do procedimento entre um pequeno número de soluções ou a convergência para uma solução que corresponda a um ótimo local. Para se evitar que isso ocorra, utiliza-se um mecanismo, denominado lista tabu, que contém as últimas alterações realizadas (movimentos de uma solução para outra). Assim, um movimento só é realizado se não pertencer a essa lista.

O procedimento descrito a seguir é uma aplicação de busca tabu ao SCP. Neste procedimento é usado como vizinhança da solução corrente o subconjunto de soluções que podem ser obtidas, a partir da corrente, com a inversão de valores de duas das variáveis. O "tabu" é fixado às trocas de valores de variáveis que levam a valores da função objetivo não menores do que o da solução corrente.

- **Heurística de busca local com lista tabu**

  1. Encontre uma solução inicial x.
  2. Faça $x^* = x$ e $Z^* = Z(x)$.
  3. Encontre $x' \in V_2(x)$, tal que $Z(x') - Z(x)$ é minimal.
  4. Se $Z(x') - Z(x) < 0$ e o movimento não é tabu:
     Faça $x = x'$;
     Atualize a lista tabu;
     Se $Z(x) < Z^*$
     Faça $x^* = x$ e $Z^* = Z(x)$;
     Retorne ao passo 3.
     senão
     Se $Z(x') < Z^*$
     Faça $x^* = x'$ e $Z^* = Z(x)$;
     Retorne ao passo 3.

3.2 Limites Inferiores

3.2.1 Heurísticas de Construção

Os limites inferiores usados nos algoritmos de Balas e Ho [BH80], Beasley [Bea87] e Fisher e Kedia [FK90], entre outros, foram obtidos a partir da resolução do problema 2.5, com o uso de uma heurística dual baseada em trabalhos anteriores ([Erl78, GS79, FH80, Won84]), possuindo a seguinte forma geral:

- **Heurística gulosa dual**

  1. $u_i = 0,\ i \in I,$
     $\Delta_i(u) = min_{j \in J_i}(c_j - \sum_{i \in I_j} u_i),\ i \in I,$
     $I(u) = \{i \in I \mid \Delta_i(u) > 0\};$
  2. Escolha $k \in I(u)$ e faça $u_k = u_k + \Delta_k(u);$
  3. Para cada $j \in J_k$
     Para cada $i \in I_j$
     Se $\Delta_i(u) > c_j - \sum_{i \in I_j} u_i$ faça:
     $\Delta_i(u) = c_j - \sum_{i \in I_j} u_i$
     Se $\Delta_i(u) = 0$ faça $I(u) = I(u) - \{i\};$
  4. Se $I(u) = \emptyset$ PARE, senão retorne ao passo 2.

O uso dessa heurística, nos três algoritmos citados anteriormente, difere apenas no critério de escolha da variável k (passo 2). Balas e Ho usaram como critério de escolha a minimização da cardinalidade do conjunto J_k, resolvendo eventuais empates com a escolha da variável de menor incremento $\Delta_k(u)$. Fisher e Kedia seguiram o mesmo caminho, diferindo apenas no critério de desempate, com a escolha da variável com o maior incremento $\Delta_k(u)$. Beasley utilizou os mesmos critérios de Balas e Ho, porém buscou o aperfeiçoamento da solução obtida com o acréscimo à heurística, após o passo 4, do seguinte ajuste nas variáveis:

5. $u_i = u_i + \min[c_j - \sum_{k \in I_j} u_k,\ j \in J)],\ i \in I.$

Existem diversas variações possíveis dessa heurística. Na implementação computacional descrita no capítulo 6, foi utilizada essa heurística, porém procurou-se chegar a uma maior uniformidade nos valores das variáveis duais. Para tanto, a escolha das variáveis tornou-se aleatória e adotou-se a estratégia de não incrementar uma variável u_k do maior valor possível $\Delta_k(u)$, sendo adicionado apenas um percentual δ desse valor. Nesse caso a linha 2 da heurística é alterada para:

2. Escolha $k \in I_u$ e faça $u_k = u_k + \delta \Delta_k u.$

3.2.2 Heurísticas de Melhoria

Uma heurística de melhoria para o problema 2.5 (dual da relaxação contínua do SCP) foi proposta por Fisher e Kedia [FK90]. Essa heurística, *3-opt*, é um procedimento de busca local análoga às descritas na seção 3.1.2 para o SCP (problema 2.1).

Nessa heurística, tenta-se a melhoria do valor da função objetivo de uma solução dual u, através de ajustes que envolvem três das variáveis duais (u_{i_1}, u_{i_2}, e u_{i_3}). A idéia é decrementar u_{i_1} de um certo valor Δ e incrementar u_{i_2} e u_{i_3} do mesmo valor, aumentando-se assim o valor da solução em Δ.

Definindo-se $J^a = \{j \in J \mid \sum_{i \in I_j} u_i = c_j\}$ e $J^i = \{j \in J \mid \sum_{i \in I_j} u_i < c_j\}$ como os conjuntos de restrições duais ativas e inativas, respectivamente, pode-se mostrar facilmente que o ajuste descrito anteriormente só é factível para $\Delta > 0$ se, e somente se, as condições abaixo são satisfeitas:

i. $u_{i_1} > 0$;

ii. $(J_{i_2} \cup J_{i_3}) \cap J^a \subseteq J_{i_1}$;

iii. $J_{i_2} \cap J_{i_3} \cap J^a = \emptyset$.

A heurística *3-opt* consiste na busca direta no conjunto $I \times I \times I$ de uma tripla i_1, i_2, i_3 que satisfaça às condições anteriores. Para a determinação do maior valor Δ para o qual é mantida a factibilidade, são identificadas as restrições inativas para as quais $\sum_{i \in I_j} u_i$ aumenta ao se proceder os ajustes em u_{i_1}, u_{i_2} e u_{i_3}. O cálculo de Δ é feito, então, como segue:

$$\begin{aligned} J^{i^*} &= \{j \in J \mid a_{i_2 j} + a_{i_3 j} - a_{i_1 j} > 0\} \\ \Delta &= \min \left\{ \min_{j \in J^{i^*}} \left[\frac{(c_j - \sum_{i \in I_j} u_i)}{a_{i_2 j} + a_{i_3 j} - a_{i_1 j}} \right], u_{i_1} \right\}. \end{aligned}$$

3.2.3 Relaxação Lagrangeana e Otimização por Subgradientes

O termo relaxação Lagrangeana foi introduzido por Geoffrion [Geo74] para especificar o problema obtido através da remoção de restrições associadas a um problema e sua inclusão na função objetivo com o uso de multiplicadores apropriados.

Relaxação Lagrangeana é baseada no fato de muitos problemas de programação inteira poderem ser vistos como um problema de fácil resolução que foi complicado por um certo conjunto de restrições. Explorando essa observação, um problema Lagrangeano é criado com a substituição desse conjunto de restrições por um termo de penalidade na função objetivo.

O marco inicial dessa abordagem é considerado como sendo os trabalhos de Held e Karp ([HK70, HK71]), embora o uso de métodos Lagrangeanos já fosse corrente antes de seu uso pelos dois autores. Em [HK70] Held e Karp mostraram que uma relaxação Lagrangeana, baseada em árvores geradoras de custo mínimo, para o TSP, fornece o mesmo limite que a relaxação contínua da formulação clássica do TSP. No segundo artigo ([HK71]) foi introduzido um método, otimização por subgradientes, para calcular os multiplicadores associados a um problema Lagrangeano. A partir de então, essas técnicas foram aplicadas com sucesso, por diferentes autores, a diversos outros problemas combinatórios.

Novos resultados e extensões aos trabalhos de Held e Karp foram obtidos por Geoffrion [Geo74], Shapiro [Sha79] e Fisher [Fis81],[Fis85] para relaxação Lagrangeana e Held, Wolfe e Crowder [HWC74], Camerini, Frata e Maffioli [CFM75] e Goffin [Gof77], para o método de otimização por subgradientes.

A primeira aplicação dessas técnicas ao SCP apareceu com o algoritmo de Etcheberry [Etc77]. Este algoritmo praticamente inaugura uma nova família de algoritmos para o SCP, onde deixa-se de usar a relaxação contínua no cálculo de limites inferiores para se adotar a relaxação Lagrangeana. Os principais algoritmos que seguem essa tendência são os de Balas e Ho [BH80], Beasley [Bea87], Fisher e Kedia [FK90] e Beasley e Jornsten [BJ92].

A abordagem adotada nos algoritmos citados anteriormente foi a total transformação do conjunto de restrições do SCP (problema 2.1) em penalidades na função objetivo, obtendo-se o seguinte problema:

$$Z_D(w) = \min \sum_{j \in J} c_j x_j + \sum_{i \in I} w_i (1 - \sum_{j \in J_i} x_j)$$
$$x_j \in \{0,1\},\ j \in J,$$

ou seja,

$$Z_D(w) = \sum_{i \in I} w_i + \sum_{j \in J} \min_{x_j \in \{0,1\}} (c_j - \sum_{i \in I_j} w_i) x_j \tag{3.1}$$

onde $w_i,\ i \in I$, são os multiplicadores associados às restrições.

Este problema é de fácil resolução e fornece um limite inferior no valor da solução ótima do problema original. Em relação a isto, Geoffrion [Geo74] mostrou que $Z_{LP} = Z_D(w^*)$ e que $Z_D(w^*) \leq Z^*,\ \forall\ w^* \geq 0$, onde w^* é o vetor de multiplicadores referente a solução ótima de 3.1, e Z_{LP} e Z^* são os valores das soluções ótimas dos problemas 2.4 e 2.1, respectivamente. Geoffrion mostrou ainda que um vetor x é solução ótima para 2.1 se, dado um certo conjunto de multiplicadores w, satisfizer as seguintes condições:

i. x é ótimo em 3.1;

ii. $\sum_{j \in J_i} x_j \geq 1,\ i \in I$;

iii. $\sum_{i \in I} w_i (1 - \sum_{j \in J_i} x_j) = 0$.

Definindo-se $\delta_j = c_j - \sum_{i \in I_j} w_i$, então claramente uma solução ótima para o problema 3.1 é obtida fixando-se:

$$x_j = \begin{cases} 1 & se\ \delta_j < 0, \\ 0\ ou\ 1 & se\ \delta_j = 0, \\ 0 & se\ \delta_j > 0. \end{cases}$$

Como $Z_D(w) \leq Z^*\ \forall\ w \geq 0$, o melhor limite inferior possível é obtido resolvendo-se $\max_{w \geq 0} Z_D(w)$. Um dos métodos para resolver esse problema é exatamente o método de otimização por subgradientes, apresentado por Held e Karp [HK71]. Este método começa com um vetor de multiplicadores w^0 e então, iterativamente, calcula direções e deslocamentos para obter uma seqüência de vetores w^k, a qual converge para o vetor w^* que maximiza $Z_D(w)$. O método pode ser sumarizado como segue:

- **Método de otimização por subgradientes**

  1. $k = 0$;
  2. Determine um vetor w^0 de multiplicadores;
  3. Resolva o problema 3.1 para w^k;
  4. Teste a condição de parada;
  5. Calcule os subgradientes $\sigma_i^k = 1 - \sum_{j \in J_i} x_j,\ i \in I$;
  6. Calcule o deslocamento $t_k = \frac{\lambda_k (Z^* - Z_D(w^k))}{\|\sigma k\|}$;
  7. Faça $w_i^{k+1} = \max(0, w_i^k + t_k \sigma_i^k)$;
  8. $k = k + 1$;
  9. Retorne ao passo 3.

Alguns pontos importantes a respeito do método :

- A escolha natural do vetor de multiplicadores w^0 é $w_i^0 = 0,\ i \in I$. Observe-se, contudo, que o método gera uma seqüência de multiplicadores w^k, onde cada vetor w^k está mais próximo do vetor ótimo w^* (em termos da norma $\|w^k - w^*\|$) que seu antecessor w^{k-1}, apesar da função objetivo não crescer monotonicamente. Portanto a convergência pode ser acelerada fazendo-se $w_i^0 = u_i,\ i \in I$, onde u é uma solução heurística do problema 2.5 (veja seção 3.2.1).

- Não havendo como provar otimalidade no método, a menos que se obtenha w^k tal que $Z_D(w^k) = Z^*$, a condição de parada é estabelecida como sendo a execução de um número limitado de iterações. Em relação a convergência do método, alguns resultados teóricos, relativos a escolha apropriada de um deslocamento t_k, mostram que $Z_D(w^k)$ converge para Z^* se satisfeitas as condições : $t_k \to 0$ e $\sum_{k=0}^{\infty} t_k = \infty$ ([HWC74]).

- Z^* pode ser substituído por um limite superior para Z_D. Este é normalmente obtido com a resolução heurística do problema 2.1 (veja seção 3.1.1).

- Alguns autores, como Beasley [Bea90], sugerem que no cálculo do deslocamento t_k o valor de Z^*, ou do limite superior adotado, seja multiplicado pelo fator 1.05, para que se previna um deslocamento muito pequeno à medida que o intervalo entre os limites inferior e superior diminui.

- A seqüência λ_k pode ser obtida fixando-se λ_0 entre 0.5 e 2 e então reduzindo-se λ_k por um fator de 2, cada vez que não houver melhoria do limite inferior em um número fixo de iterações. Este esquema, embora largamente utilizado na prática, viola a condição de que $\sum t_k = \infty$. Assim, há a possibilidade de convergência para um ponto que não pertença ao espaço de soluções.

- As justificativas para o modo como são calculados t_k e λ_k podem ser encontradas em [HK71, HWC74, Gof77].

Beasley [Bea90] propôs um algoritmo heurístico baseado na aplicação das técnicas descritas nesta seção. O algoritmo utiliza as técnicas de relaxação Lagrangeana e otimização por subgradientes para gerar limites inferiores no valor da solução ótima para uma instância do SCP. Agregado ao método dos subgradientes, após cada iteração do passo 7, soluções factíveis são obtidas através da seleção de colunas para as quais as restrições duais associadas foram violadas e completando-as, se necessário, de forma gulosa (veja seção 3.1.1).

Capítulo 4

Algoritmos para Resolução do *SCP*

4.1 Introdução

A maioria dos algoritmos que foram propostos para a resolução do SCP são baseados em metodologias enumerativas. Muitos desses algoritmos combinam ao esquema de enumeração diversas outras técnicas, tais como: regras de pré-processamento, heurísticas primais e duais, relaxação Lagrangeana e otimização pelo método dos subgradientes, planos de cortes e outras.

Embora a maioria dos algoritmos apresente essa característica de combinar diferentes técnicas, eles podem ser classificados em três grupos principais, de acordo com as técnicas dominantes ou mais importantes ([CK75, SM89]): algoritmos enumerativos, baseados em planos de cortes e heurísticos.

No capítulo 3 foram descritos alguns dos principais algoritmos heurísticos para a resolução do SCP. As principais características de alguns algoritmos dos outros dois grupos, enumerativos e baseados em planos de cortes, são descritas no restante deste capítulo. Aplicações de técnicas enumerativas ao SCP são os algoritmos de Lawler [Law66], Pierce [Pie68], Garfinkel e Nemhauser [GN69], Lemke, Salkim e Spielberg [LSS71], Pierce e Lasky [PL73], Guha [Guh73], Etcheberry [Etc77], Beasley [Bea87] e Fisher e Kedia [FK90]. Os principais algoritmos para resolução do SCP, baseados em planos de cortes, foram desenvolvidos por House, Nelson e Rado [HNR66], Bellmore e Ratliff [BR71], Balas e Ho [BH80] e Beasley e Jornsten [BJ92], podendo ser citados ainda Salkin e Koncal [SK73], que descreveram em um estudo computacional a aplicação do algoritmo de Gomory [Gom63] ao SCP.

4.2 Algoritmos Enumerativos

Uma importante classe de algoritmos para resolução de problemas de programação inteira consiste de métodos de busca. A forma mais simples desses métodos equivale a enumeração exaustiva de todas as soluções. As formas mais elaboradas são acrescidas de técnicas que permitem a enumeração de apenas uma parte do conjunto de soluções, com o descarte de soluções não promissoras.

Contudo, a eficiência de tais algoritmos depende enormemente da habilidade de enumerar implicitamente a maior parte das soluções. As principais metodologias enumerativas incluem as técnicas de enumeração implícita e técnicas de *branch and bound*. O primeiro tipo é mais adequado a problemas de programação inteira 0-1, e pode ser considerado como um caso especial de *branch and bound* [Tah75].

No caso do SCP, os algoritmos considerados mais promissores ou que apresentaram melhores resultados são baseados em técnicas de *branch and bound*. Dos algoritmos citados na seção anterior encontram-se nessa classe os de Lemke, Salkin e Spielberg, Etcheberry, Balas e Ho, Beasley e Fisher e Kedia, dentre outros.

As técnicas de enumeração implícita baseiam-se em dois princípios básico: um esquema enumerativo que garanta que todas as soluções são enumeradas (implícita ou explicitamente) de modo não redundante e regras de descarte que excluam o maior número possível de soluções parciais não promissoras.

Branch and Bound são esquemas enumerativos fundamentados em duas operações básicas. A primeira é a decomposição do problema original em subproblemas. A segunda operação envolve o cálculo de limites inferiores e superiores no valor da função objetivo. O objetivo dessa operação é acelerar o processo de descarte de subproblemas que não podem gerar soluções promissoras, diminuindo, conseqüentemente, a enumeração.

A noção de decomposição é importante, pois fornece um esquema simples de enumeração das soluções para o problema tratado. Essa enumeração é construída com a decomposição do problema original em subproblemas, e a decomposição recursiva destes. Com isto, o espaço de soluções é particionado se forem observadas as seguintes regras:

1. Qualquer solução factível para o problema original deve ser solução factível para exatamente um de seus subproblemas;

2. Uma solução factível para qualquer um dos subproblemas deve ser factível para o problema original.

Considerando-se que o número de subproblemas que podem ser gerados é muito grande, embora finito, o sucesso dessa estratégia depende do descarte do máximo de subproblemas, evitando-se assim a total enumeração do conjunto de soluções.

Nesse contexto a utilização de relaxações do problema original é fundamental, pois bons critérios de descarte de subproblemas podem ser obtidos observando-se as seguintes propriedades:

R1 - Se a relaxação não tem solução factível, o mesmo é verdadeiro para o problema original;

R2 - O valor da solução ótima do problema original não é menor do que o valor da solução ótima da relaxação (considerando-se um problema de minimização);

R3 - Se uma solução ótima da relaxação é factível para o problema original, então é solução ótima também para este.

A própria definição de relaxação implica essas propriedades, cujas demonstrações são triviais. Tomando-se, por exemplo, a relaxação contínua (problema 2.4) do SCP (problema 2.1), tem-se que essas propriedades são conseqüências diretas das descritas na seção 2.4.

De acordo com as propriedades R1, R2 e R3 anteriores, um subproblema não será decomposto em novos subproblemas, e será conseqüentemente descartado, se a análise de uma relaxação mostrar que o subproblema:

- não tem solução factível (decorrente de R1); ou,
- não tem solução factível de valor objetivo menor do que a melhor solução conhecida para o problema original (decorrente de R2); ou,
- é satisfeito pela solução ótima para a relaxação (decorrente de R3).

4.2.1 Algoritmo de Lemke, Salkin e Spielberg (1971)

Nas propriedades descritas na seção 2.4, foi mostrado que uma solução factível para o SCP (problema 2.1) pode ser construída a partir de uma solução factível para a relaxação contínua do mesmo (problema 2.4). O algoritmo de *branch and bound*, proposto por Lemke, Salkin e Spielberg [LSS71], é baseado na resolução da relaxação linear do SCP (obtenção de limites inferiores) e na construção de soluções para o SCP (obtenção de limites superiores).

Inicialmente a relaxação linear (problema 2.4) associada ao subproblema corrente é resolvida. Se o subproblema é infactível ou o valor da função objetivo excede o melhor limite superior (Z_{UB}), então o problema é descartado. Caso contrário, é guardado o índice j^* referente à variável de menor valor fracionário na solução ótima e é verificado se o arredondamento desta solução tem valor inferior a Z_{UB}. Neste caso é obtida a solução para o subproblema associado às variáveis fracionárias (propriedades 3, 6 e 7, seção 2.4).

O uso do método *dual-simplex* para resolver a relaxação do subproblema pode fornecer um bom critério para o descarte do mesmo. Observe que o problema 2.5

apresenta solução dual viável inicial (propriedade 12, seção 2.4). Assim, sendo este um problema de maximização, a seqüência de valores da função objetivo gerada pelo método *dual-simplex* é estritamente crescente. Portanto, o subproblema pode ser descartado se em qualquer iteração do método, o valor da função objetivo (Z_{LP}) exceder o valor do melhor limite superior conhecido (Z_{UB}).

Após a resolução da relaxação, caso a solução obtida não seja inteira, então, depois da extração de uma solução factível para o SCP (obtenção do limite Z_{UB}), o problema é decomposto em subproblemas. O processo de decomposição baseia-se na escolha de uma restrição $i \in I_{j^*}$, tal que a cardinalidade $|J_i|$ seja mínima. Um novo subproblema é construído a partir da escolha de um $j \in J_i$ que satisfaça às condições:

i. x_j satisfaça o maior número possível de restrições não satisfeitas, ao menor custo unitário;

ii. o custo c_j de x_j, mais o custo das outras variáveis já fixadas em 1, não exceda Z_{UB} (melhor limite superior obtido).

4.2.2 Algoritmo de Etcheberry (1977)

Este algoritmo de enumeração implícita ([Etc77]) segue a estrutura básica descrita anteriormente. A grande inovação contida nesse algoritmo é a aplicação da relaxação Lagrangeana e do método de otimização por subgradientes ao SCP. Essa abordagem praticamente inaugura uma nova classe de algoritmos, já que a maioria dos que o seguem utilizam as mesmas técnicas.

A exemplo da maioria dos algoritmos para resolução do SCP, este também utiliza algumas regras de redução do tamanho do problema (veja seção 2.3). Essas são aplicadas tanto ao problema original quanto aos diversos subproblemas advindos do processo de decomposição.

O esquema de decomposição utilizado é similar ao esquema de particionamento de linhas, proposto por Marsten [Mar74] em seu algoritmo para resolução do SPP. A estratégia é baseada na escolha de restrições i_1 e i_2, tal que $J_{i_1} \cap J_{i_2} \neq \emptyset$ e $J_{i_1} - J_{i_2} \neq \emptyset$. Dois novos subproblemas (SCP_1 e SCP_2) são então obtidos trocando-se:

- as restrições i_1 e i_2 pela restrição $\sum_{j \in J_{i_1} \cap J_{i_2}} x_j \geq 1$; ou,
- a restrição i_1 por $\sum_{j \in J_{i_1} - J_{i_2}} x_j \geq 1$ e $x_j = 0 \ \forall \ j \in J_{i_1} \cap J_{i_2}$.

Definindo-se $\cup D_t$ como a união de todos os subconjuntos mutuamente disjuntos de $J - (J_{i_1} \cap J_{i_2})$, a decomposição de SCP_1 e SCP_2 é dependente de tais conjuntos D_t. A regra usada para selecionar um subproblema da lista de candidatos é LIFO sendo que, a cada decomposição, SCP_2 é armazenado antes de SCP_1.

Escolhido um subproblema, a escolha de uma restrição i é feita de tal modo que a interseção de J_i com $\cup D_t$ seja mínima. Com essa estratégia é provável que o limite inferior associado ao subproblema em questão seja significativamente melhorado. O critério de descarte de um subproblema é baseado no limite inferior obtido com à aplicação do método dos subgradientes a relaxação Lagrangeana do subproblema.

O algoritmo utiliza duas estratégias diferentes na seleção de um vetor inicial de multiplicadores Lagrangeanos w, a cada vez que o método dos subgradientes for aplicado a um subproblema. A primeira estratégia é a manutenção do último vetor w disponível, ou seja, aquele resultante do processamento do subproblema anterior ao corrente. A segunda é a utilização do vetor w relativo ao subproblema do qual o corrente foi decomposto. Aparentemente as duas estratégias são iguais. Contudo, lembrando-se que o processo de decomposição gera sempre dois novos problemas (SCP_1 e SCP_2, com SCP_2 sendo armazenado primeiro) e a regra para selecionar um problema é LIFO, então nota-se que as estratégias diferenciam-se no processamento de SCP_2.

4.2.3 Algoritmo de Beasley (1987)

Este algoritmo ([Bea87]) combina à metodologia de *branch and bound*, testes de redução do tamanho do problema, uma abordagem heurística para resolução do dual da relaxação linear do SCP (problema 2.5) e programação linear.

No algoritmo é utilizada a heurística gulosa proposta por Balas e Ho [BH80] (veja seção 3.1.1) para obtenção de limites superiores e um procedimento constituído de três etapas, baseado em otimização por subgradientes e programação linear, para gerar limites inferiores.

A primeira etapa do procedimento de cálculo de limites inferiores consiste na aplicação de uma heurística gulosa dual, descrita na seção 3.2.1, para resolver o dual da relaxação linear do SCP (problema 2.5), gerando assim valores iniciais para os multiplicadores da relaxação Lagrangeana associada ao problema. Na segunda, é usado o método dos subgradientes para tentar melhorar o limite inferior obtido na etapa anterior. A terceira etapa consiste na resolução do problema 2.5 até a otimalidade, com o uso do método simplex.

A eficácia do procedimento anterior depende muito da aplicabilidade de testes de redução do tamanho do problema, conforme descrito na seção 2.3, e de algumas penalidades simples que forçam uma determinada variável a pertençer, ou não, a uma solução.

Uma das penalidades é baseada nos coeficientes da relaxação Lagrangeana. Considerando-se $Z_D(w)$ como o limite inferior associado a um determinado conjunto w de multiplicadores Lagrangeanos, Z_{UB} como o melhor limite superior e $C_j = c_j - \sum_{i \in I_j} w_i,\ j \in J$. Tem-se que, se $Z_D(w) + C_j > Z_{UB}$ $(C_j \geq 0)$, então a variável x_j pode ser eliminada do problema, pois a sua inclusão na

solução levaria o limite inferior a ultrapassar o limite superior. Similarmente, se $Z_D(w) - C_j > Z_{UB}$ ($C_j < 0$), então a variável x_j deve pertencer à solução ótima.

Duas outras penalidades, baseadas na alteração do custo de uma variável, foram usadas na tentativa de redução do tamanho do problema. Na primeira, fixa-se em "infinito" o custo c_j de uma variável e aplica-se a heurística gulosa dual, com qualquer vetor u inicial válido. Se o limite obtido exceder Z_{UB}, então a variável x_j deve pertencer à solução ótima. Na segunda, fixam-se em zero as variáveis duais u_i tal que $i \in I_j$, para determinado $j \in J$. Se o limite inferior obtido com a aplicação da heurística dual, somado ao custo c_j, exceder Z_{UB}, então a variável j não pode pertencer à solução ótima.

Outra penalidade utilizada é baseada no particionamento de uma determinada restrição i. Nesse caso definem-se dois conjuntos J^1 e J^2, tal que $J^1 \cup J^2 = J_i$ e $J^1 \cap J^2 = \emptyset$, e fixam-se em "infinito" os custos c_j, $j \in J^1$. Se o limite inferior obtido com a aplicação da heurística dual exceder Z_{UB}, então pelo menos uma variável $j \in J^1$ deve pertencer à solução ótima. Logo, pode-se fazer $J_i = J^1$. Em relação a essa penalidade Beasley observou que a desigualdade $\sum_{j \in J^1} x_j \geq 1$ pertence à família de cortes definida por Balas [Bal80]. Assim, essa penalidade pode ser vista como um caso especial da abordagem de planos de cortes utilizada por Balas e Ho [BH80].

No cálculo de um limite inferior para o problema original, são executadas as três etapas do procedimento de cálculo. No final da terceira etapa, resolução da relaxação linear do SCP, caso a solução ótima encontrada não seja inteira, o problema é decomposto em outros subproblemas. Para estes são executadas apenas as duas primeiras etapas do procedimento.

O procedimento de decomposição dos subproblemas é baseado na escolha de restrições, ainda não satisfeitas, e na fixação em 1 de variáveis dessas restrições. A escolha da restrição depende da classe do problema a ser tratado. Se o problema for de custos não unitários, então a escolha da restrição baseia-se nos valores dos multiplicadores Lagrangeanos resultantes da aplicação do método dos subgradientes. Nesse caso é escolhida a restrição i cujo valor do multiplicador w_i associado seja mínimo. Se o problema for de custos unitários, o critério é a minimização de $|J_i|$. Escolhida a restrição, um novo subproblema é criado fixando-se em 1 uma variável $j \in J_i$, tal que o valor $c_j - \sum_{i \in I_j} w_i$ seja mínimo.

Um problema é descartado quando o limite inferior, calculado pela resolução da relaxação Lagrangeana, exceder o limite superior, ou quando os testes de redução indicarem que o problema não tem solução factível.

- **Algoritmo de Beasley**

  1. $Z_{LB} = -\infty; \quad Z_{UB} = +\infty;$
  2. Utilize testes lógicos para redução do tamanho do problema.

3. Utilize a heurística gulosa dual para encontrar uma solução factível para o dual da relaxação linear do problema (limite Z_{LB}).
4. Utilize a heurística gulosa primal para encontrar uma solução factível para o problema (limite Z_{UB}).
5. Utilize o método dos subgradientes.
6. Reutilize os testes lógicos para redução do tamanho do problema.
7. Utilize o método *simplex* para resolver o dual da relaxação linear do problema. Atualize Z_{LB}.
8. Se a solução dual obtida é inteira, vá para o passo 10.
9. Decomponha o problema corrente em novos subproblemas.
10. Se não existem subproblemas ativos, PARE. O recobrimento associado a Z_{UB} é ótimo.
11. Escolha um novo subproblema.
12. Utilize as heurísticas e o método dos subgradientes para tentar melhorar os limites Z_{LB} e Z_{UB}.
13. Se $Z_{LB} \geq Z_{UB}$ descarte o problema corrente e retorne ao passo 10.
14. Retorne ao passo 9.

4.2.4 Algoritmo de Fisher e Kedia (1990)

O algoritmo de Fisher e Kedia [FK90] é o primeiro a tratar um modelo misto SCP/SPP. Este modelo inclui como casos especiais cada um dos problemas em separado.

Uma característica importante desse algoritmo é a utilização de heurísticas contínuas no dual (problema 2.5) da relaxação contínua do SCP (problema 2.4), para obtenção de limites inferiores a serem usados em um procedimento de *branch and bound*. A motivação dessa abordagem é o fato de que a relaxação contínua, principalmente do SPP, pode ser altamente degenerada e de difícil resolução.

O algoritmo é composto de uma heurística gulosa contínua para obtenção de uma solução inicial para o problema 2.5 (veja seção 3.2.1); uma heurística, *3-opt*, que tenta melhorar a solução obtida com a heurística anterior através de ajustes em conjuntos de três variáveis duais (seção 3.2.2); o método dos subgradientes (seção 3.2.3); uma heurística gulosa (seção 3.1.1) que constrói uma solução para o SCP, utilizando a solução dual na seleção de valores para as variáveis primais; testes lógicos para verificação de factibilidade e redução de tamanho do problema (seção 2.3); e um procedimento de *branch and bound*.

O eixo principal do algoritmo é a aplicação das várias heurísticas para obtenção de limites inferiores e superiores. Caso o intervalo entre estes seja superior a um determinado percentual, tenta-se melhorar o limite inferior através da

aplicação do método dos subgradientes à relaxação Lagrangeana obtida a partir do problema 2.5.

No caso em que os limites não sejam iguais, o problema é decomposto em subproblemas, o que é feito a partir da escolha de uma restrição i e da fixação de variáveis $x_j \in J_i$ em 1. Pelo fato do algoritmo trabalhar com um modelo misto SCP/SPP, o processo de decomposição depende do tipo da restrição escolhida, ou seja, se $i \in I^1$ (SCP) ou $i \in I^2$ (SPP). Os dois casos são:

- se a restrição $i \in I^1$, a decomposição depende do valor do custo reduzido de cada variável $j \in J_i$. Se $c_j - \sum_{i \in I_j} u_i > Z_{UB} - Z_{LB}$, então o problema que seria criado fixando-se $x_j = 1$ pode ser descartado, pois não geraria solução ótima. Caso contrário, considerando-se que os custos reduzidos estão ordenados de forma crescente segundo $j_1, \ldots, j_{|J_i|}$, os novos subproblemas são criados fixando-se $x_{j_k} = 1$ e $x_{j_1} = \ldots = x_{j_{k-1}} = 0$, deixando-se as restantes livres.

- se $i \in I^2$, então $|J_i|$ novos subproblemas são criados fazendo-se $x_j = 1$, $\forall\, j \in J_i$.

Para cada novo subproblema é aplicado o conjunto de heurísticas primais/duais, atualizando-se o valor da melhor solução primal, se for o caso. Um subproblema é descartado se for infactível ou o limite superior, correspondente à melhor solução primal, não for maior que o limite inferior obtido com a aplicação das heurísticas duais.

O processo de decomposição gera uma lista de subproblemas para posterior tratamento. A cada passo de escolha de um subproblema, é escolhido aquele com a melhor solução factível, e o esquema descrito anteriormente é aplicado a ele:

- **Algoritmo de Fisher e Kedia**

  1. $Z_{LB} = -\infty; \quad Z_{UB} = +\infty;$
  2. Utilize testes lógicos para redução do tamanho do problema.
  3. Utilize as heurísticas duais (gulosa e 3-*opt*) para encontrar soluções factíveis para o dual do problema corrente.
  4. Utilize a heurística gulosa primal para encontrar uma solução factível para o problema corrente.
  5. Se o intervalo entre as soluções primal e dual for superior a 1%, utilize o método dos subgradientes e reutilize a heurística gulosa primal.
  6. Atualize, se for o caso, Z_{LB} e Z_{UB}.
  7. Se $Z_{UB} \leq Z_{LB}$ descarte o problema corrente e vá para o passo 9.
  8. Decomponha o problema corrente em novos subproblemas.
  9. Se ainda existirem subproblemas ativos, escolha um novo subproblema e retorne ao passo 2.
  10. PARE. O recobrimento associado a Z_{UB} é ótimo.

4.3 Algoritmos Baseados em Planos de Cortes

Os métodos de planos de cortes baseiam-se no fato de que uma solução ótima, para um problema de programação linear não inteira, encontra-se em um ponto extremo do espaço de soluções. O princípio dos algoritmos baseados em tais métodos, para problemas de programação linear inteira, é que novas restrições, violadas pela solução ótima para a relaxação linear do problema, sejam apropriadamente escolhidas e adicionadas ao problema. Isto deve ser feito de tal modo que o espaço de soluções seja progressivamente reduzido, até que a solução inteira ótima coincida com um ponto extremo.

O termo "planos de cortes" é sugerido pelo fato de que essas restrições dividem o espaço de soluções, eliminando regiões que não contêm soluções factíveis para o problema em questão.

4.3.1 Algoritmo de Bellmore e Ratliff (1971)

Este algoritmo ([BR71]) utiliza técnicas de planos de cortes de forma diferente do exposto acima. Os cortes utilizados aqui não se limitam a excluir soluções infactíveis para o problema. Os cortes são construídos de forma que soluções factíveis previamente consideradas não possam ser encontradas novamente. Isto é conseguido estipulando-se que o valor da função objetivo de uma nova solução factível deva ser inferior ao da melhor solução conhecida (de todas as conhecidas).

Esta idéia para construção dos cortes é baseada nas propriedades descritas na seção 2.4, onde é mostrado que uma solução ótima para o SCP (problema 2.1) deve ser um ponto extremo para a relaxação linear do mesmo (problema 2.4).

Portanto, se x é um ponto extremo de 2.4, B e C_B são a matriz correspondente a base associada a x e o respectivo vetor de custos, então a relação entre Z (o custo de x) e o custo Z' de uma solução qualquer x' de 2.4 pode ser expressa como $Z' - Z = -\sum_{j=1}^{n+m}(\sum_{i=1}^{m} C_B B^{-1} a_{ij} - c_j)x_j$.

Segue-se então uma condição necessária (mas não suficiente) para que $Z' - Z < 0$ (e conseqüentemente a solução x possa ser melhorada) :

$$\sum_{j \in Q} a_{ij} x'_j \geq 1, \tag{4.1}$$

onde $Q = \{j \mid \sum_{i=1}^{m} C_B B^{-1} a_{ij} - c_j > 0\}$. Se $Q = \emptyset$ implica que x é uma solução ótima para 2.4 e também para 2.1. Isto baseia-se nos fundamentos do método *simplex*, ou seja, para se encontrar uma solução melhor deve-se seguir por uma das arestas do poliedro que levam a soluções de menor valor.

Deve-se notar que o conjunto Q não contém índices referentes às variáveis de folga. Logo, a desigualdade 4.1 é exatamente da mesma forma das restrições de 2.1. Note-se ainda que a determinação do conjunto Q é mais simples do que parece, não sendo necessárias operações de inversão de matrizes. Isto deve-se à propriedade de involução da matriz B, descrita na seção 2.4.

O algoritmo descrito a seguir é baseado no acréscimo sucessivo de uma restrição da forma 4.1. Na descrição do algoritmo, SCP_0 corresponde ao problema 2.1 e SCP_k é o problema resultante do acréscimo de k cortes ao problema original. A forma geral do algoritmo é a seguinte:

- **Algoritmo de Bellmore e Ratliff**

  1. $SCP_0 = SCP$;
     $Z^* = \infty$, $x^* = \emptyset$;
     $k = 0$.
  2. Seja x^k uma solução não redundante para SCP_k e B uma correspondente base com propriedade de involução.
     Se $\sum_{j=1}^{n} c_j x_j^k < Z^*$, então atualize Z^* e x^*.
  3. Seja $Q = \{j \mid \sum_{i=1}^{m} C_B B^{-1} a_{ij} - c_j > 0\}$.
     Se $Q = \emptyset$, então PARE X^* é solução ótima.
  4. Acrescente a restrição $\sum_{j \in Q} a_{ij} x_j \geq 1$ a SCP_k.
     $k = k + 1$.
     Retorne ao passo 2.

Um limite superior no número de recobrimentos é dado por 2^n, com isso mostra-se facilmente que o algoritmo pára após um número finito de iterações, pois pelo menos um recobrimento é eliminado a cada iteração.

O algoritmo de Bellmore e Ratliff pode ser visto como uma evolução do proposto por House, Nelson e Rado [HNR66]. Nesse algoritmo, uma nova restrição é adicionada ao problema, caso a solução da relaxação linear do SCP (problema 2.4) não corresponda a um ponto extremo integral. Essa restrição não elimina a melhor solução encontrada até então para o SCP e ainda garante que, pelo menos, mais uma variável x_j, da solução de 2.4, terá valor fixado em 1. O mesmo processo é repetido para o novo problema e recursivamente para os seguintes, até que se obtenha um problema cuja solução inteira corresponda a um ponto extremo da relaxação linear do mesmo.

4.3.2 Algoritmo de Balas e Ho (1980)

A abordagem de Balas e Ho [BH80] é importante, tanto teoricamente quanto na prática, devido ao fato de combinar ao esquema de "*branch and bound*" um conjunto de heurísticas primais e duais, otimização por subgradientes, relaxação Lagrangeana e planos de cortes derivados de limites condicionais.

A idéia central dessa abordagem é a derivação de desigualdades válidas para o SCP a partir de limites inferiores condicionais. Esses limites são válidos se certas desigualdades, também chamadas condicionais, tiverem sido adicionadas ao conjunto de restrições.

A transposição de um limite superior por um tal limite inferior condicional indica que qualquer solução, de valor menor que o limite superior, viola pelo menos uma das desigualdades condicionais. Isto fornece uma disjunção válida que pode ser usada para particionar o conjunto de soluções, ou para derivar uma família de planos de cortes.

Para se obter planos de cortes a partir de limites condicionais é necessária uma solução para o dual da relaxação linear do SCP (problema 2.5). Por outro lado, para se garantir que os planos de cortes gerados são todos distintos, cada desigualdade deve eliminar algum recobrimento que satisfaça a todas as desigualdades previamente geradas. Portanto, para se obter uma seqüência de planos de cortes distintos, também é necessária uma seqüência de recobrimentos.

O algoritmo proposto baseia-se na intercalação de dois procedimentos para obtenção dessas duas seqüências. O primeiro procedimento gera um recobrimento x para o problema corrente. O segundo gera uma solução dual e usa-a para derivar uma desigualdade que elimina x. O algoritmo converge na proporção que o intervalo $Z_{UB} - Z_{LB}$, entre os limites obtidos pelos dois procedimentos, decresce. Quando a convergência torna-se muito lenta, o problema é decomposto em novos subproblemas. O algoritmo pode ser descrito como segue:

- **Algoritmo de Balas e Ho**

  1. Use heurísticas para encontrar soluções factíveis para o problema corrente (limite superior Z_{UB}) e para o dual da relaxação linear deste (limite inferior Z_{LB}).
  2. Utilize testes apropriados para fixar variáveis em 0.
  3. Se $Z_{LB} \geq Z_{UB}$ descarte o problema corrente e vá para o passo 10.
     Se alguma variável pertencente ao recobrimento corrente foi fixada em 0, retorne ao passo 1.
  4. Gere um plano de corte violado pelo recobrimento corrente.
  5. Se o número de iterações não é múltiplo de α retorne ao passo 1.
  6. Utilize o método dos subgradientes para tentar melhorar a solução dual e o limite inferior.
  7. Reutilize os testes de fixação de variáveis em 0.
  8. Se $Z_{LB} \geq Z_{UB}$, descarte o problema e vá para o passo 10.
     Se o intervalo $Z_{UB} - Z_{LB}$ diminuiu de pelo menos $\epsilon > 0$ durante as últimas β iterações, retorne ao passo 1.
  9. Decomponha o problema corrente em novos subproblemas.
  10. Se ainda existem subproblemas ativos, escolha um novo subproblema (usando a regra LIFO) e retorne ao passo 1.
  11. PARE. O recobrimento associado a Z_{UB} é ótimo.

Os autores utilizaram valores para α, β e ε definidos com base em experimentos númericos (α é a frequência de aplicação do método dos subgradientes). Estes parâmetros foram fixados em $|J|/10 \leq \alpha \leq |J|/20$, $\beta = 4\alpha$ e $\epsilon = 0.5$.

Na obtenção de limites inferiores e superiores no valor da solução ótima para um problema, são utilizados no algoritmo dois conjuntos de heurísticas duais e primais. Esses dois conjuntos são compostos de heurísticas que, basicamente, são variações das descritas nas seções 3.1.1 e 3.2.1.

O método dos subgradientes usado no algoritmo é uma especialização do procedimento geral descrito na seção 3.2.3. Uma diferença em relação ao procedimento geral é o uso de uma relaxação que utiliza desigualdades definidas por uma família de subconjuntos disjuntos. Nesse caso é escolhido um subconjunto (maximal) $I' \subset I$ tal que $J_i \cap J_k = \emptyset \ \forall \ i, k \in I' \ (i \neq k)$, e definidas as desigualdades:

$$\begin{aligned} \textstyle\sum_{j \in J_i} x_j &\geq 1, \quad i \in I' \\ \textstyle\sum_{j \in J} d_j x_j &\geq d_0 \\ 0 \leq x_j &\leq 1, \quad j \in J \end{aligned}$$

onde $d_j = |I_j| - 1$ se $j \in J_i$ para algum $i \in I'$, ou $d_j = |I_j|$ caso contrário, e $d_0 = |I - I'|$. A utilização de desigualdades definidas por conjuntos disjuntos baseia-se em idéia similar utilizada por Etcheberry [Etc77].

O algoritmo utiliza a rotina a seguir, a qual implementa um procedimento proposto em [Bal80], para gerar um corte válido. Neste procedimento x é o recobrimento corrente, u uma solução para o dual da relaxação contínua do problema corrente, $S(x) = \{j \in J \mid x_j = 1\}$, $T(x) = \{i \in I \mid \sum_{j \in J_i} x_j = 1\}$, Z_{UB} e Z_{LB} os valores das soluções x e u, respectivamente, e $s_j = c_j - \sum_{i \in I_j} u_i$, $j \in J$.

- **Construção de cortes condicionais**

  1. Seja $W = \emptyset$,
     $S = \{j \in S(x) \mid s_j > 0\}$,
     $t = 1$;
  2. Seja $v_t = \min\{\max_{j \in S} s_j, \min_{j \in S}\{s_j \mid s_j \geq Z_{UB} - Z_{LB}\}$,
     $Q = \{j \in J \mid s_j \geq v_t\}$,
     $J' = \{j \in S \mid s_j = v_t\}$,
     $I_{J'} = \bigcup_{j \in J'} I_j$;
     Escolha $i(t) \in T(x)$ tal que $|J_{i(t)} \setminus Q \cup W| = \min_{i \in T(x) \cap I_{J'}} |J_i \setminus Q \cap W|$;
     Seja $J^t = J' \cap J_{i(t)}$;
     Faça $W = W \cup (J_{i(t)} \setminus Q)$,
     $Z_{LB} = Z_{LB} + \sum_{j \in J^t} s_j$;
     Se $Z_{LB} < Z_{UB}$, faça
     $S = S \setminus J^t$,

$$s_j = \begin{cases} s_j - \sum_{j \in J^t} s_j & se\ j \in Q \cap J_k \\ s_j & caso\ contrário \end{cases}$$

$t = t + 1$,

retorne ao passo 1.

3. Adicione ao SCP a desigualdade : $\sum_{j \in W} x_j \geq 1$.

Este procedimento termina depois de um número de iterações igual a cardinalidade de $S(x)$, ou seja, igual ao número de $j \in S(x)$, tal que $s_j > 0$. O resultado é uma desigualdade satisfeita por todos os recobrimentos melhores do que o recobrimento x associado a Z_{UB}, e violada por este. Esta mesma propriedade está associada aos cortes de Bellmore-Ratliff (veja seção 4.3.1), os quais são obtidos com o uso de bases com propriedade de involução. Balas [Bal80] mostrou que esses cortes constituem uma subclasse dos cortes condicionais.

Na fixação de variáveis em zero foi adotado o critério de se fixar todas as variáveis $j \in J$, tal que $s_j \geq Z_{UB} - \sum_{i \in I_j} u_i$. Este critério é utilizado a cada vez que uma nova solução dual é obtida, seja pelas heurísticas ou pelo método dos subgradientes.

O procedimento de decomposição do problema em subproblemas utiliza, intermitentemente, duas regras. A primeira decompõe o problema de acordo com a disjunção $\vee_{i=1}^{p}(x_j = 0,\ j \in Q_i \mid \sum_{j \in Q_k} x_j \geq 1,\ k = 1, \ldots, i-1)$, onde os conjuntos Q_i são construídos de modo similar ao usado no procedimento que gera os cortes. A segunda regra é uma variação da utilizada por Etcheberry [Etc77]. Aqui são escolhidos $i, k \in I$ $(i \neq k)$, tal que i corresponde ao último corte adicionado ao problema, com $|J_k \cap J_i| = \min_{J_h \cap J_i \neq \emptyset} |J_h \cap J_i|$ e então o problema é decomposto de acordo com a disjunção $(x_j = 0,\ j \in J_i \cap J_k) \vee (\sum_{j \in J_i \cap J_k} x_j \geq 1)$.

4.3.3 Algoritmo de Beasley e Jornsten (1992)

Este algoritmo ([BJ92]) é uma evolução do proposto anteriormente por Beasley (veja seção 4.2.3). Foram realizadas algumas alterações e introduzidas novas técnicas ao mesmo.

Uma alteração ocorreu no cálculo de limites superiores para a solução ótima do SCP. Inicialmente era utilizada uma heurística gulosa, proposta por Balas e Ho [BH80], para gerar soluções factíveis para o SCP. Neste novo algoritmo foi utilizada uma heurística Lagrangeana proposta por Beasley [Bea90] (veja seção 3.2.3, a qual, segundo os autores, apresentou melhores resultados para SCP de custos não unitários.

Essa heurística considera os coeficientes x da relaxação Lagrangeana do SCP (veja equação 3.1) como uma solução parcial para este. Para se chegar a uma solução completa, depois de identificadas as restrições não satisfeitas, são adicionadas à solução as variáveis de menor custo necessárias ao recobrimento dessas

restrições. O último passo consiste na transformação dessa solução em outra não redundante.

Além dessa alteração foi introduzida uma inovação. Foram incluídas antes da resolução da relaxação linear, restrições que excluem uma solução factível. Considerando-se que x é uma solução factível para o SCP e $J' = \{j \in J \mid x_j = 1\}$, então $J' - \{j\}$ $(j \in J')$ não é solução factível (x é não redundante). Logo, adicionando-se ao problema as restrições $\sum_{j \in J'} x_j \leq |J'| - 1$ e $\sum_{j \notin J'} x_j \geq 1$, x é excluída do conjunto de soluções do problema, o qual torna-se infactível se esta for a única solução ótima.

Outra inovação foi o uso de cortes de Gomory, na tentativa de melhorar o limite inferior obtido com a resolução da relaxação linear do SCP. Neste caso, a relaxação do problema já com as restrições que excluem uma solução factível. A estratégia utilizada na aplicação dos cortes de Gomory foi a seguinte:

a. foram gerados 30 cortes a partir da solução da relaxação linear e então o novo problema foi resolvido usando o método *dual-simplex*;

b. a metade dos cortes foram gerados a partir das variáveis x_j, da solução ótima da relaxação, com maiores valores fracionários;

c. os cortes restantes foram gerados a partir de variáveis de folga, também as de maiores valores fracionários.

O processo de decomposição de um problema é baseado na escolha de uma restrição i, não satisfeita, tal que o produto $w_i\sigma_i$ seja máximo (w_i e σ_i são o multiplicador de Lagrange e o subgradiente, respectivamente, associados à restrição i após da aplicação do método dos subgradientes à relaxação Lagrangeana). A decomposição é feita a partir da escolha de uma variável $j \in J_i$, tal que o valor de x_j, da solução da relaxação linear obtida após a adição dos cortes de Gomory, seja máximo.

Capítulo 5

Abordagem proposta

5.1 Introdução

Ao considerar a classe NP-difícil de problemas, observa-se que, para um populoso espectro de problemas dessa classe, são conhecidos algoritmos aproximados que fornecem soluções de boa qualidade para um certo conjunto de instâncias do problema atacado. Mais ainda, quando o estudo concentra-se em casos particulares de um problema, algoritmos exatos e "rápidos" podem existir. Por outro lado, estes algoritmos podem falhar, seja por não se adequarem à instância, seja pela dificuldade de se identificar que a instância corresponde a um caso particular.

Um problema para o qual nenhum algoritmo conhecido é completamente satisfatório motiva a proposição de métodos que empreguem uma variedade de algoritmos. Estes métodos multi-algorítmicos procuram utilizar diversos algoritmos simples para realizar o trabalho que antes caberia a um único algoritmo complexo. A idéia é que, para problemas com este grau de dificuldade, algoritmos simples possam cooperar de modo a gerar soluções que se equiparem, ou mesmo superem, a solução gerada por um algoritmo complexo. Neste contexto, algoritmos simples são em geral heurísticas que implementam uma intuição sobre o problema, e têm como característica básica um tempo reduzido de processamento.

A simples idéia de combinar diferentes técnicas para a resolução de um problema, entretanto, não é suficiente para obter um algoritmo que garanta uma eficiência aceitável na resolução de qualquer instância do problema. Bons exemplos dessa dificuldade são os métodos exatos descritos no capítulo 4. Embora a maioria desses métodos seja multi-algorítmica, observando-se os seus resultados computacionais, nota-se que o comportamento deles, frente a uma mesma classe de instâncias, pode diferir muito.

A principal dificuldade está na interação entre diferentes técnicas, pois nem sempre uma determinada combinação destas permite uma boa utilização do tempo disponível para o processamento, ou mesmo uma colaboração eficiente entre as mesmas.

Estas dificuldades são reduzidas quando utiliza-se uma arquitetura como a de Times Assíncronos. Estes são organizações de *softwares* propostas por Talukdar e Souza ([TS90, Tal93, ST93, Sou93]) que visam a interação eficiente entre vários algoritmos para a resolução de problemas adequados à abordagem multi-algorítmica.

Uma experiência do emprego deste paradigma é o trabalho desenvolvido por Souza [Sou93], que encontrou as soluções ótimas de várias instâncias da versão Euclideana do TSP com até várias centenas de cidades. Souza utilizou algoritmos heurísticos tradicionais (*Lin-Kernighan*, *Or-opt*, *Arbitrary Insertion* e *Held-Karp*), bem como alguns algoritmos simples especialmente desenvolvidos para integrar os Times Assíncronos utilizados. Duas outras experiências são os trabalhos desenvolvidos por Peixoto ([Pei95]) e Cavalcante ([Cav95]), que aplicaram este paradigma ao *Flow Shop Problem* e ao *Job Shop Scheduling*, respectivamente.

5.2 Times Assíncronos

Uma organização é um grupo de indivíduos ou agentes que cooperam entre si, seguindo regras e estruturas pré-definidas, na busca de um objetivo comum. As organizações, como modelos de ação, podem ser classificadas em diversos grupos, por exemplo, segundo a flexibilidade das regras de cooperação entre seus agentes.

As organizações, em particular as de "*softwares*", foram classificadas por Talukdar e Souza [TS90, TS92, TS93, Tal93] segundo um modelo estrutural. De acordo com este modelo, uma organização é um esquema no qual agentes combinam-se para formar super-agentes. Neste modelo a estrutura interna de um agente não é de nenhum interesse. Deste modo, apenas são considerados os super-agentes cujos sistemas de comunicação (o mecanismo pelo qual os agentes comunicam-se entre si) possam ser modelados através de um número finito de memórias compartilhadas.

Nesse contexto, super-agentes são coleções de memórias compartilhadas por seus agentes. Assim, a estrutura de um super-agente caracteriza-se, principalmente, pelos fluxos de dados e fluxos de controles. Os fluxos de dados determinam a quais memórias um agente pode ter acesso. Os fluxos de controles definem quais informações os agentes podem ler/escrever nas memórias, quando os agentes podem processar essas informações e com que recursos.

Em uma das configurações possíveis nesse modelo estrutural, há a ocorrência de fluxos cíclicos de dados e inexistem fluxos de controle nos super-agentes. Este caso é definido como sendo um *A-Team* (do inglês *Asynchronous Team*). Portanto, um *A-Team* é composto por agentes e memórias compartilhadas, os quais satisfazem as seguintes condições:

- Todos os agentes em um *A-Team* atuam de forma autônoma, não existindo controle de dependência entre eles (no contexto aqui estudado agentes são

algoritmos com um protocolo de comunicação);

- Os fluxos de dados são cíclicos, ou seja, os dados de saída de um agente são potencialmente dados de entrada para outro agente ou ele mesmo;
- A comunicação entre agentes é assíncrona e se faz exclusivamente através de memórias compartilhadas de dados.

As experiências citadas anteriormente com esse paradigma sugerem que a cooperação entre os agentes tende a ser sinérgica, ou seja, o resultado produzido por estes, quando olhados em conjunto, pode ser melhor do que a da soma dos resultados obtidos isoladamente. Espera-se então uma maior chance de se encontrar uma solução próxima à ótima global para o problema sendo tratado pelo *A-Team*. Essas experiências sugerem também que um *A-Team* é uma organização eficiente em escala, ou seja, o seu desempenho melhora com a introdução de novos elementos (agentes e memórias).

A característica mais atraente do paradigma de *A-Team* está na sua flexibilidade. Os agentes em um *A-Team*, atuando de forma autônoma, ficam suscetíveis de serem introduzidos ou retirados da organização a qualquer momento. Outro ponto importante é que a comunicação entre os agentes sendo assíncrona permite que estes sejam executados em paralelo, com a comunicação entre eles gerando um contínuo fluxo de modificações no conteúdo das memórias compartilhadas.

5.2.1 Representação Gráfica

Uma possível representação gráfica para *A-Teams* é o uso de arcos para representar agentes e retângulos para memórias compartilhadas. Desse modo, o direcionamento dos arcos indica leitura (de onde partem) e escrita (onde chegam) nas memórias, e o uso de retângulos permite a composição de memórias para formar uma nova memória compartilhada.

Um exemplo de *A-Team*, composto de seis agentes e duas memórias, é ilustrado na figura 5.1. Neste exemplo o agente A_1 lê dados na memória M_1 e escreve na memória M_2, A_2 lê e escreve em M_2, A_3 lê dados em ambas e escreve em M_1 e o agente A_4 lê em M_2 e escreve em M_1 ou M_2 (ou em ambas). O agente D_1 elimina dados não relevantes das memórias M_1 e M_2 e o agente D_2 age do mesmo modo na memória M_2.

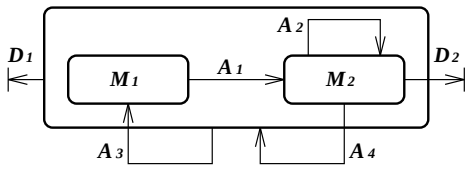

Figura 5.1: Exemplo de representação gráfica de *A-Teams*

Usando-se esta representação gráfica tornam-se evidentes as três características básicas dos *A-Teams*. A condição dos fluxos de dados serem cíclicos, por exemplo, é bastante clara na figura 5.1. Existem aí vários ciclos, como o formado pela memória M_1, o agente A_1, a memória M_2 e o agente A_4.

5.2.2 Parâmetros de Projeto

Talukdar e Souza [TS92] definiram como eficientes em escala os super-agentes cujas arquiteturas são abertas (novos elementos, agentes e memórias, podem ser introduzidos ou retirados a qualquer momento) e cujos membros, novos e velhos, cooperam eficazmente entre si. Uma das hipóteses levantadas pelos dois autores, em relação aos *A-Teams*, é que o subconjunto de *A-Teams* que contém mecanismos de busca de consenso na população de dados ali armazenados, tem uma grande interseção com o subconjunto de super-agentes eficientes em escala.

A sinergia em super-agentes, como definida por Talukdar [Tal93], é o fato de que os resultados obtidos por um super-agente são maiores (melhores) do que a soma dos resultados obtidos pelos seus componentes em separado. Em relação a isto, Talukdar observou que a própria estrutura básica dos *A-Teams* contribui para um comportamento anárquico do mesmo (agentes autônomos podem agir com propósitos conflitantes). Contudo, levantou a possibilidade de se evitar tal comportamento com o uso de estratégias adequadas, tornando assim, sinérgica a cooperação entre os componentes do *A-Team*.

Os *A-Teams* são, portanto, capazes de conseguir eficiência em escala e sinergia. Contudo, o processo como isso se dá não é totalmente claro. Dois fatores que contribuem para tal são:

- a manutenção de uma população de dados alternativos aumenta a chance de que alguma dessas alternativas gere bons resultados;
- a eliminação de alternativas ruins é, no mínimo, tão importante para o progresso do *A-Team* quanto a geração de boas alternativas.

Souza [Sou93], baseando-se nas hipóteses expostas e também em resultados empíricos, definiu os critérios listados a seguir como os mais importantes a serem observados no projeto de um *A-Team*:

- **Fluxos de dados :** a estrutura de um *A-Team* está diretamente ligada com os algoritmos disponíveis para a construção dos agentes. Estes, e seus relacionamentos com as memórias, devem ser projetados de modo a se aproveitar ao máximo qualquer informação a respeito dos algoritmos, para que o desempenho do *A-Team* e a qualidade das soluções geradas sejam os melhores possíveis;
- **Algoritmos de consenso :** estes algoritmos geram novas soluções para o problema que está sendo tratado, incorporando nesse processo informações

(partes) de soluções já armazenadas nas memórias. Procura-se, deste modo, aproveitar as diferenças e semelhanças entre estas soluções, aumentando a eficiência da busca de novas soluções através da restrição do espaço de busca das mesmas;

- **Estratégias de iniciação das memórias :** no preenchimento inicial das memórias deve-se buscar um equilíbrio entre a alta diversidade das soluções geradas e sua qualidade. Uma alta diversidade pode significar um grande número de soluções de baixa qualidade, as quais seriam processadas inutilmente pelos agentes. Na busca deste equilíbrio, uma memória pode ser preenchida, por exemplo, com soluções geradas aleatoriamente, ou com soluções produzidas por algoritmos de construção, ou ainda, combinando-se estas duas abordagens. Outro ponto importante é que o preenchimento total não é necessário, uma vez que as piores soluções podem vir a ser descartadas logo em seguida, sem que nenhum agente as tenha selecionado. Deve-se, portanto, preencher as memórias com um número mínimo de soluções que garanta um bom funcionamento do *A-Team*;

- **Políticas de seleção de dados :** a política de seleção define de que modo e de quais memórias os agentes lêem os dados que processam. Na escolha dos dados, para um certo agente, deve-se observar primeiro se este é determinístico, pois neste caso o processamento de um mesmo dado duas vezes será inútil. Assim, a escolha dos dados deve obedecer uma rígida seqüência para evitar o desperdício de recursos. No caso de agentes não determinísticos, pode-se escolher os dados segundo alguma distribuição de probabilidade, uma vez que o processamento repetido de um mesmo dado pode gerar resultados diferentes;

- **Políticas de destruição :** a política de destruição define a estratégia para a troca ou exclusão de dados nas memórias. A estratégia a ser adotada é tão importante quanto a geração de novos dados, já que a exclusão de dados não promissores influencia diretamente a diversidade dos dados nas memórias, auxiliando-se, assim, a convergência e direcionamento dos dados ali armazenados. Uma boa estratégia considera que algumas características dos dados armazenados (qualidade, número de vezes que foi lido, etc.) podem ser usadas para associar a cada um uma probabilidade de ser excluído da memória. Considerando-se uma memória contendo soluções para um determinado problema, pode-se adotar, entre outras políticas, a eliminação da pior solução com probabilidade 1, ou a destruição de qualquer solução com distribuição uniforme (ou linear) de probabilidade, ou ainda a combinação destas políticas, conforme a diversidade das soluções na memória; e

- **Tamanho das memórias :** este critério é importante uma vez que a taxa

de diversidade do conteúdo de uma memória está diretamente ligada ao seu tamanho. Adotando-se um tamanho muito pequeno de memória, os dados ali armazenados podem convergir muito rapidamente. Por outro lado, em memórias muito grandes, a convergência pode ser muito lenta, além de se correr o risco de subutilização das mesmas. Nos dois casos, o desempenho do *A-Team* é influenciado negativamente.

Estes critérios relacionados acima foram organizados por Peixoto [Pei95], em uma seqüência de passos e sugestões, formando uma metodologia de especificação de Times Assíncronos para problemas de Otimização Combinatória de uma função objetivo.

5.3 Arquiteturas Gerais de *A-Teams* para o *SCP*

Os algoritmos descritos no capítulo 4 conseguiram bons resultados para muitas instâncias do SCP, solucionando grandes instâncias do mesmo até a otimalidade, ou bem perto disso, em no máximo alguns minutos. Contudo, quase todos os algoritmos falharam na obtenção de boas soluções, para uma ou outra instância, em um limite razoável de tempo. Além disso, são conhecidas hoje instâncias do SCP consideradas de difícil resolução e para as quais esses algoritmos são considerados inadequados.

A tais instâncias, normalmente, estão associados certos fatores, como por exemplo um grande intervalo entre os valores das soluções ótimas para a instância e a correspondente relaxação contínua (*gap* de integralidade); a densidade elevada dos subconjuntos, ou seja, um grande número de elementos por subconjunto em relação ao total de elementos; e distribuições muito particulares dos elementos nos subconjuntos. Além destes fatores, os custos associados aos subconjuntos também influenciam na facilidade/dificuldade de resolução de uma instância.

Considerando-se estas dificuldades, faz-se necessário definir quais informações se deseja extrair de tais instâncias. Embora existam várias opções, para diversas situações reais é suficiente considerar os seguintes objetivos a serem alcançados com a resolução do SCP:

a. obtenção de limites superiores no valor da solução ótima;

b. obtenção de limites inferiores e superiores; ou

c. obtenção de limites inferiores e superiores, e garantia de otimalidade de uma solução.

A seguir são propostos modelos gerais de *A-Teams* para a resolução do SCP, visando cada um dos objetivos citados. Na próxima seção é proposto um *A-Team*, de acordo com o terceiro objetivo, para resolução do SCP.

- ***A-Team* básico**

  O objetivo desse *A-Team* básico é a geração de limites superiores, no valor da solução ótima, para uma determinada instância do SCP, ou seja, utilizam-se apenas soluções primais. Para que este intuito seja alcançado de forma satisfatória, o modelo é composto de quatro componentes básicos enumerados a seguir:

  a. uma memória para armazenamento de soluções factíveis para o problema a ser tratado;
  b. um agente iniciador que constrói soluções factíveis e armazena-as na memória, preenchendo-a total ou parcialmente;
  c. um ou mais agentes modificadores de soluções, os quais atuam sobre o conteúdo da memória, modificando e aprimorando as soluções ali contidas;
  d. um agente que elimina, segundo algum critério, soluções da memória, permitindo, assim, que novas soluções possam ser inseridas na mesma.

  Um modelo simples para esse *A-Team* básico é mostrado na figura 5.2. Deve-se notar que o modelo mostrado não é único, assim como não os são os outros propostos a seguir. Contudo, é suficiente para exemplificar um *A-Team* para o SCP que cumpra satisfatoriamente o objetivo em questão (geração de limites superiores).

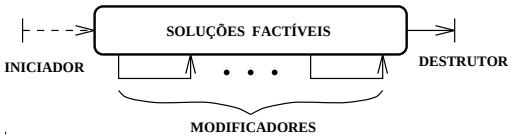

Figura 5.2: Arquitetura básica de *A-Team* para o SCP

- ***A-Team* primal-dual**

  O *A-Team* básico pode ser estendido adotando-se uma abordagem primal-dual no tratamento do problema a ser resolvido. Com esta abordagem pode-se desenvolver, conjuntamente, limites inferiores e superiores para o valor da solução ótima. O objetivo deste modelo é garantir que a melhor solução encontrada esteja a um certo percentual da solução ótima.

  A figura 5.3 mostra uma extensão do modelo básico. Além dos componentes descritos anteriormente, foram incorporados ao modelo os seguintes itens:

  a. uma memória para armazenamento de soluções factíveis para uma relaxação do SCP (ou para o dual da relaxação), ou seja, limites inferiores;

b. agentes que constroem soluções para o SCP, utilizando soluções da relaxação como auxílio nesse processo;

c. um agente destrutor para agir na memória de soluções relaxadas, com atuação similar ao já descrito para a outra memória;

d. um agente iniciador do conteúdo da memória de soluções relaxadas, também similar ao descrito para a outra memória;

e. agentes modificadores de soluções, que atuam em cada uma das memórias, ou em ambas.

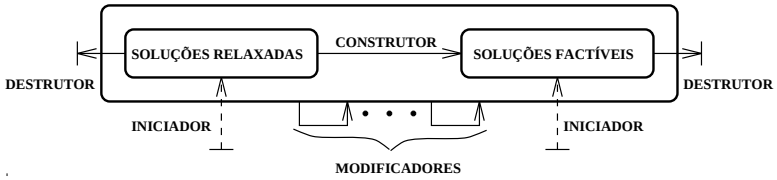

Figura 5.3: Arquitetura primal-dual de *A-Team* para o SCP

- ***A-Team* primal-dual + cortes**

  Um resultado bem conhecido da teoria de programação linear é o que diz respeito ao intervalo, que pode existir, entre os valores das soluções ótimas para o problema original e uma relaxação do mesmo. Tendo-se em vista este resultado, o objetivo do modelo anterior pode ser ainda expandido, buscando-se, além de bons limites inferiores e superiores, a garantia de otimalidade de uma solução. Para isto, o modelo deve ser expandido de modo a comportar novas desigualdades válidas para o problema, as quais, uma vez adicionadas ao mesmo, consigam diminuir o intervalo entre os limites, permitindo assim, eventualmente, provar a otimalidade de uma solução.

  Os novos componentes deste novo modelo, mostrado na figura 5.4, são:

  a. uma memória para armazenamento de novas desigualdades válidas para o problema;

  b. um agente, ou mais, responsável pela construção de cortes para o problema.

  Além do acréscimo desses dois novos componentes, ocorre ainda a adaptação dos outros componentes, descritos no modelo anterior, para que levem em conta as novas desigualdades.

5.4 Aplicação de *A-Teams* ao *SCP*

Para se testar a aplicabilidade de *A-Teams* na resolução do SCP, procurou-se observar os três objetivos de qualidade das soluções, descritos na seção 5.3.

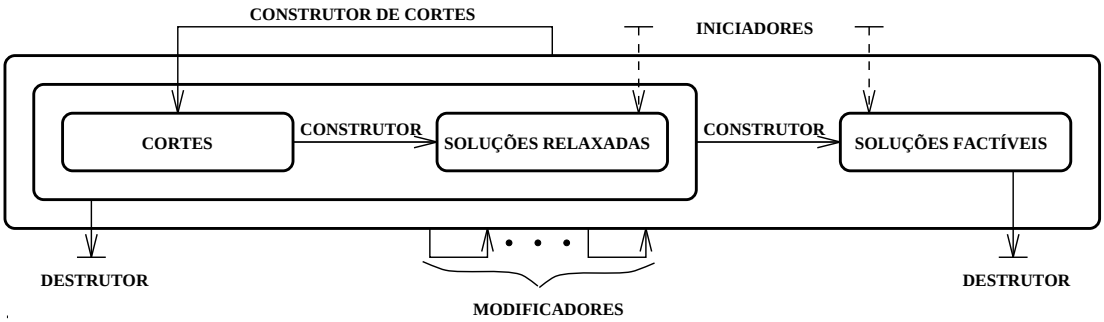

Figura 5.4: Arquitetura primal-dual de *A-Team*, com utilização de cortes, para o SCP

A arquitetura proposta a seguir permite que qualquer um deles seja alcançado. Isto é conseguido simplesmente com a inclusão, ou exclusão, de componentes ao *A-Team*, de modo a deixá-lo com a arquitetura de um dos três modelos descritos anteriormente.

O *A-Team* mostrado na figura 5.5 tem a arquitetura do modelo primal-dual + cortes. Contudo, é fácil observar que eliminado-se alguns componentes e adaptando-se outros, pode ser modificado para se encaixar nas arquiteturas dos outros dois modelos. O *A-Team* é composto de três memórias compartilhadas e dez agentes. Compõem, ainda, a estrutura do *A-Team*, agentes que preenchem inicialmente as memórias e agentes que mantêm a população de soluções dentro de um determinado nível. Outro elemento usado é a composição de memórias. É importante observar que os algoritmos (e técnicas heurísticas) que compõem os agentes foram descritos no capítulo 3.

Em conformidade com o modelo primal-dual + cortes, busca-se aqui a determinação de bons limites inferiores e superiores no valor da solução ótima e a quebra de eventuais intervalos entre estes limites. Para o cálculo de limites inferiores, optou-se pela resolução do dual da relaxação contínua do SCP (problema 2.5), motivado pela disponibilidade de bons métodos heurísticos para tal. Nas subseções a seguir, descreve-se essa parte do *A-Team* e, em seguida, os outros componentes.

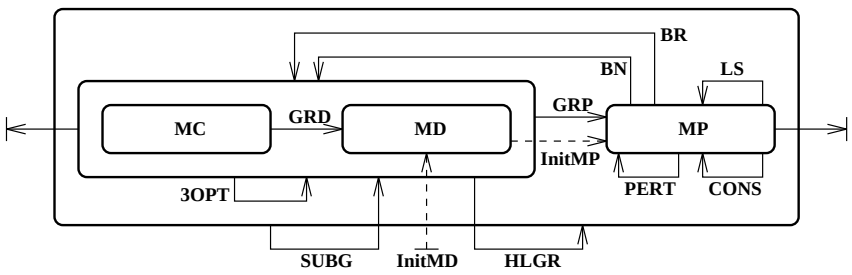

Figura 5.5: Proposta de arquitetura geral de A-Team para a resolução do SCP

5.4.1 Memórias

A estrutura do *A-Team* é composta de três memórias compartilhadas básicas e duas outras memórias que são composições das primeiras. Observe que as duas últimas são virtuais, uma vez que seu uso é somente no intuito de simplificar a representação da interação dos agentes com as básicas. As memórias básicas são:

MP : armazena soluções factíveis para a instância do SCP que está sendo tratada (problema 2.1), ou seja, soluções cujos valores são limites superiores no valor da solução ótima.

MD : armazena soluções factíveis para o problema dual da relaxação contínua do SCP (problema 2.5), ou seja, limites inferiores. Esta memória é dividida em duas partes, uma de tamanho fixo correspondente às variáveis duais do problema original e uma parte de tamanho variável, que pode ser alocada dinamicamente, correspondente às variáveis duais criadas com a adição de cortes ao problema original.

MC : armazena novas restrições válidas (cortes) para a instância em questão.

5.4.2 Iniciadores das Memórias

Os iniciadores das memórias são agentes especiais responsáveis pela geração de uma população inicial de soluções e preenchimento das memórias. Apenas as memórias MD e MP sofrem a ação destes agentes. A terceira memória é preenchida no decorrer da execução do time como um todo. Isto é devido as estratégias adotadas para a construção de novas restrições (veja seção 5.4.3). Essas estratégias dependem da existência de bons limites inferiores e superiores, nem sempre disponíveis com o simples preenchimento das memórias MD e MP. Logo, os iniciadores são:

InitMD : Iniciador da memória de soluções duais. Este agente utiliza um procedimento baseado em uma heurística gulosa dual, descrita na seção 3.2.1, para gerar as soluções. Este procedimento difere da heurística em dois aspectos. O primeiro é o não determinismo, ou seja, ao contrário do que ocorre na heurística, a ordem na qual as variáveis duais são escolhidas, para terem os valores incrementados, é aleatória. O segundo aspecto, é o uso intercalado, a cada execução do procedimento, de uma das duas políticas de cálculo do incremento das variáveis duais, descritas na seção 3.2.1. O objetivo destas modificações é garantir o máximo de diversidade nas soluções geradas.

InitMP : Iniciador da memória de soluções primais. Este agente também busca o preenchimento da memória com a máxima diversidade possível entre as soluções geradas. Neste processo é utilizada uma heurística gulosa, descrita

na seção 3.1.1, alternando-se, a cada execução da mesma, a função de escolha das variáveis para compor uma solução. Quando a função a ser utilizada for a proposta por Fisher e Kedia [FK90], o agente busca na memória de soluções duais as informações necessárias ao uso de tal função.

5.4.3 Agentes

Os agentes classificam-se em quatro tipos, segundo as memórias para as quais fornecem o resultado de sua execução, ou em outra palavras, segundo o tipo de solução ou estruturas que geram:

1. Agentes duais - geram soluções para o dual da relaxação contínua do SCP. São eles:

   GRD : Construtor de soluções duais. A função deste agente é a construção de novas soluções duais e envio das mesmas para a memória correspondente. Seu funcionamento é basicamente igual ao do agente InitMD. A diferença é que aqui, antes de se gerar uma nova solução, busca-se, na memória de cortes, novas restrições que são acrescentadas ao problema.

   SUBG : Modificador de soluções duais. Este agente utiliza o método de otimização por subgradientes (seção 3.2.3) para a resolução da relaxação Lagrangeana descrita pela equação 3.1. O agente busca informações, necessárias ao método dos subgradientes, nas três memórias: um limite superior no valor da solução ótima do SCP é obtido na memória de soluções primais; novas restrições são buscadas na memória de cortes; e da memória de soluções duais é obtida uma solução, cujos valores das variáveis duais funcionam como os multiplicadores Lagrangeanos iniciais. A nova solução dual, obtida com a resolução da relaxação Lagrangeana, é enviada pelo agente para a memória de soluções duais.

   3-OPT : Modificador de soluções duais. A ação deste agente consiste na busca de uma solução na memória de soluções duais, e na tentativa de aprimoramento da mesma, com o uso da heurística *3-opt* descrita na seção 3.2.2. Sendo o resultado, uma nova solução dual, devolvido para a mesma memória.

2. Agentes primais - responsáveis pela geração de soluções factíveis para a instância do SCP que está sendo tratada. São eles:

   GRP : Construtor de soluções primais. Este agente é basicamente igual ao agente InitMP, com a diferença que é considerada a possibilidade de adição de novas restrições ao problema. Neste caso, se necessário, o agente busca as novas restrições na memórias de cortes.

LS : Modificador de soluções primais. Este agente busca uma solução na memória de soluções primais e tenta melhorá-la com o uso de uma heurística de busca local (veja seção 3.1.2) baseada em busca tabu (Glover [Glo89, Glo90]). No caso, é usado como vizinhança da solução corrente o subconjunto de soluções que podem ser obtidas, a partir da corrente, com a inversão de valores de duas das variáveis. O "tabu" é fixado às trocas de valores de variáveis que levam a valores da função objetivo não menores do que o da solução corrente.

PERT : Modificador de soluções primais. O objetivo deste agente é o aprimoramento de uma solução primal através da perturbação desta. O processo de perturbação consiste na transformação da solução em outra redundante, o que se dá com a escolha aleatória de novas variáveis a serem adicionadas à solução. A nova solução é, então, obtida com o descarte do máximo de variáveis, da solução perturbada, de modo que o recobrimento não seja afetado. Este processo é repetido um certo número de vezes, para uma mesma solução inicial, e a melhor solução obtida nas várias iterações é armazenada na memória de soluções primais. Esta heurística está descrita na seção 3.1.2.

CONS : Modificador/Construtor de soluções primais. A ação deste agente consiste na determinação da interseção (variável a variável) entre três soluções armazenadas na memória de soluções primais. O esquema adotado aqui é uma especialização da heurística de consenso descrita na seção 3.1.2. Considerando-se que x^1 é a melhor solução ali armazenada, x^2 e x^3 as outras duas, e x^0 a nova solução a ser gerada, então o critério de consenso utilizado pelo agente é: $x_j^0 = 1$ se $x_j^1 = x_j^2 = 1$ ou $x_j^1 = x_j^3 = 1$, caso contrário $x_j^0 = 0$ ($j \in J$). Essa solução parcial pode não corresponder a um recobrimento. Neste caso é completada com as mesmas heurísticas gulosas (e mesmos critérios) utilizadas no agente GRP.

3. Agente híbrido (dual/primal) - é um agente que gera simultaneamente soluções para o SCP e para o dual da relaxação linear deste.

HLGR : Modificador de soluções duais e construtor de soluções primais. Este agente é baseado na heurística Lagrangeana proposta por Beasley [Bea90]. Essa heurística consiste na aplicação do método dos subgradientes a uma relaxação Lagrangeana do SCP, com a geração de soluções factíveis para o SCP a cada iteração do método. Aqui, como no agente SUBG, são necessários uma solução dual e um limite superior no valor da solução ótima, para a execução do método dos subgradientes. Estas informações são buscadas nas memórias adequadas, levando-se em conta ainda, o fato de que a solução dual pode

ter sido gerada após a adição de novas restrições ao problema, neste caso, estas são buscadas na memória de cortes. A solução dual, obtida ao final do método dos subgradientes, e a melhor solução primal, das obtidas em cada iteração do método, são armazenadas nas memórias correspondentes.

4. Agentes que geram desigualdades válidas e desigualdades condicionalmente válidas. Neste segundo caso são gerados cortes, a partir de uma determinada solução, os quais eliminam esta e outras soluções de menor valor objetivo. Assim, um corte deste tipo sempre elimina pelo menos uma solução válida. Portanto, sob a condição de que nehuma solução melhor do que a que gera o corte é eliminada, diz-se que a desigualdade é válida condicionalmente neste caso.

   BR : Este agente gera desigualdades do tipo descritas por Bellmore e Ratliff [BR71]. O agente busca na memória MP uma solução e gera novas desigualdades utilizando o esquema mostrado na seção 4.3.1.

   BN : As desigualdades geradas por este agente foram descritas por Balas e Ng [BN89a]. O procedimento de construção destas é mostrado na seção 2.5. A partir da definição de um conjunto $S \subset I$, somam-se as desigualdades $i \in S$ e ajustam-se os coeficientes a_{ij} da nova desigualdade, tal que $a_{ij} \in \{0, 1, 2\}$. Para a definição desse conjunto S, o agente busca uma solução primal na memória MP e considera este conjunto como sendo aquele formado pelas desigualdades saturadas (soma das variáveis igual a 1).

5.4.4 Políticas de Seleção e Destruição de Soluções

No modelo proposto foram consideradas três políticas de acesso às memórias. Seguindo essas políticas, os agentes selecionam (na memória MC) cortes aleatoriamente e selecionam uma solução (na memória MD ou MP): aleatoriamente (distribuição uniforme de probabilidade); com distribuição linear de probabilidade da melhor para a pior solução; ou, também com distribuição linear, da pior para a melhor. As mesmas políticas foram consideradas na escolha de soluções a serem eliminadas das memórias MD e MP.

A política de destruição adotada para a memória MC, na eliminação de um corte, considera o número de soluções duais aos quais tal corte esteja relacionado. Assim, só são eliminados cortes que não estão associados a soluções da memória MD.

A motivação para a adoção dessa política é a facilidade no controle das variáveis duais, ou seja, ao se eliminar um corte não há necessidade de se verificar nas diversas soluções contidas na memória MD a ocorrência de uma variável dual que corresponda ao corte em questão. Espera-se que essa política, juntamente

com a eliminação de soluções duais de menor qualidade, mantenha na memória os cortes mais fortes, ou seja, aqueles que podem vir a contribuir para a redução do *gap* de integralidade.

Capítulo 6

Resultados Computacionais

6.1 Implementação de *A-Teams*

A implementação do *A-Team* proposto seguiu uma arquitetura cliente-servidor, que torna mais fácil o processamento paralelo/distribuído. Nesta estrutura, os agentes funcionam como clientes e as memórias como servidores de dados. Assim, os clientes (agentes) requisitam informações a processos que atuam como servidores (memórias), através de chamadas a procedimentos remotos. Alguns fatores foram determinantes na escolha desta estrutura:

1. a inexistência de restrições quanto ao número de clientes (agentes) e servidores (memórias);

2. a garantia de integridade dos dados nas memórias, decorrente do tratamento, pelo servidor, das requisições dos clientes segundo a política FIFO;

3. a existência de várias ferramentas de suporte a essa abordagem (por exemplo o DPSK+P desenvolvido por Cardozo e Sichman [Car87, CS92].

O código foi desenvolvido na linguagem C, utilizando as facilidades de programação e suporte a aplicações distribuídas oferecidas pelo DPSK+P. Esta ferramenta é composta basicamente de um *kernel* responsável pelas funções de processamento distribuído, uma interface para conexão do código desenvolvido à ferramenta e um conjunto de facilidades para gerenciar e inspecionar os objetos ativos (clientes e servidores).

Usando-se este modelo de implementação todos os algoritmos (clientes e servidores) podem ser executados concorrentemente em um computador (com um ou mais processadores) ou distribuídos em uma rede, com um computador alocado para cada algoritmo. Isso sem esforço adicional de programação (alteração de código).

6.2 Definição de uma Configuração

No desenvolvimento deste trabalho foram realizados exaustivos testes computacionais. Nestes testes foram analisadas diversas configurações de *A-Teams* baseadas no modelo geral *primal-dual + cortes* proposto na seção 5.3. As configurações testadas são subconjuntos da arquitetura geral para a resolução do SCP proposta na seção 5.4. A configuração representada na figura 6.1 foi a que apresentou melhores resultados e foi adotada nos testes descritos nas próximas seções.

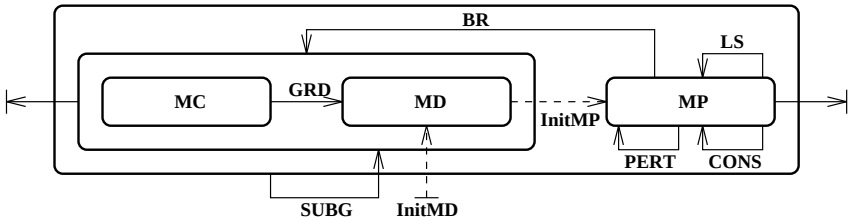

Figura 6.1: Configuração do *A-Team* utilizada nos testes computacionais

Para a escolha de tal configuração, os testes consistiram na execução das configurações escolhidas com um conjunto restrito de instâncias (1.3, 2.1 e 3.3 - veja seção 6.3). Determinou-se um conjunto de parâmetros (tamanho de memórias, número máximo de cortes a ser adicionado ao problema e tempo máximo de execução do time) e realizaram-se para cada instância quatro testes, em média, em cada conjunto de valores dos parâmetros. O tamanho das memórias variou de poucas dezenas de elementos (em torno de 30) a algumas centenas (de 600 a 700), o número de cortes ficou entre menos de uma dezena e algumas centenas (de 300 a 400) e o tempo de processamento variou de 1800 segundos a 170.000 segundos (48 horas).

Um dos principais objetivos dos testes realizados foi a análise do comportamento de cada um dos agentes em relação a um certo subconjunto de agentes, ou seja, qual o tempo de processamento necessário para se gerar uma solução, qual a qualidade desta solução e como o agente em questão depende da geração de boas soluções pelos outros agentes. No caso da configuração que foi adotada nos testes finais, os seus agentes mostraram-se dominantes em relação aos demais. A respeito do comportamento dos agentes, são relevantes as seguintes observações:

3-OPT : este agente apresenta melhores resultados ao trabalhar com soluções duais nas quais existam muitas variáveis com folga. Assim, os agentes GRD e SUBG o dominaram completamente, pois os métodos heurísticos utilizados nestes geram soluções com um grande número de variáveis com folga muito pequena ou nula.

GRP : o comportamento deste agente é dominado pelo do agente CONS. As heurísticas de construção de soluções são as mesmas utilizadas no agente

CONS. Contudo, neste último as heurísticas são utilizadas para completar soluções parciais, o que levou a melhores resultados.

HLGR : este agente pode ser eliminado das configurações testadas, pois as soluções duais geradas equiparavam-se às geradas pelo agente SUBG, enquanto as soluções primais geradas mostraram-se inferiores, para a maioria das instâncias testadas, às geradas pelos agentes LS, PERT e CONS.

BN : os cortes gerados por este agente mostraram-se mais fracos do que os gerados pelo agente BR. Um dos motivos foi que o procedimento utilizado no agente BN, para construção das desigualdades, não garante que estas definam facetas.

Os testes preliminares serviram também para a definição de alguns critérios, para a implementação da configuração mostrada na figura 6.1. Baseando-se nos testes, escolheram-se tamanhos e políticas de iniciação e de destruição diferenciadas para cada uma das memórias, e ainda, políticas de seleção mais adequadas a cada um dos agentes.

A conclusão a que se chegou, em relação ao tamanho das memórias, foi que o desempenho não é sensível a pequenas variações. Os testes mostraram que, para uma mesma instância, pode-se chegar a resultados praticamente iguais usando-se diferentes tamanhos de memórias. Contudo, observou-se também que os melhores resultados foram obtidos para valores em determinadas faixas. Considerando-se $|M\alpha|$ como o tamanho da memória α e δ o menor valor entre $|J|$ e $|I|$, as conclusões para cada memória são as seguintes:

MD : $|MD| \in [\delta, 2\delta]$. Para valores abaixo desta faixa, a diversidade de soluções na memória cai muito rapidamente, limitando o espaço de busca. Adotando-se valores superiores, observou-se a subutilização das memórias;

MP : $|MP| \in [2\delta, 4\delta]$. A escolha de valores abaixo, e acima, desta faixa implica na diminuição, e aumento, respectivamente, da diversidade na memória; e

MC : Aparentemente devido à diversidade das instâncias testadas e à estrutura de dados adotada na implementação desta memória (tabela *hash*), não se conseguiu determinar uma faixa de tamanhos que representassem um bom equilíbrio entre o total de cortes armazenados e a eficiência de acesso. Contudo, observou-se para $|MC| < 10\delta$ um comportamento totalmente insatisfatório do *A-Team*.

A estratégia de iniciação das memórias MD e MP, que rendeu melhores resultados, foi o preenchimento de apenas 40% a 60% da capacidade das mesmas. Para valores abaixo de 40%, ocorreram problemas relativos à diversidade no conteúdo das memórias. Adotando-se valores acima de 60%, foi observado que muitas das soluções eram descartadas sem ser utilizadas pelos agentes.

Determinaram-se, a partir dos testes realizados, políticas de destruição diferenciadas para as memórias MD e MP. A estratégia que apresentou melhores resultados para a memória MP foi a escolha e exclusão de qualquer solução, exceto a melhor delas, com distribuição linear de probabilidade, crescente da melhor para a pior.

A definição de uma estratégia para a memória MD foi um pouco mais complexa, devido ao comportamento dos agentes GRD e SUBG e à necessidade de manutencão da diversidade na memória. Observou-se que as soluções geradas pelos dois agentes situam-se em patamares completamente distintos. O agente SUBG gera, a partir de soluções geradas por GRD, soluções com valores objetivos bastante próximos (ou mesmo superiores, se considerados os cortes) ao valor da solução ótima para a relaxação contínua do problema. Assim, a estratégia de destruição considera qual foi o agente gerador da nova solução a ser incluída. Quando GRD envia uma solução para a memória, é excluída outra com distribuição linear de probabilidade, crescente da melhor para a pior. Para o outro agente (SUBG) é utilizada a mesma política de escolha da solução a ser excluída, porém com probabilidade crescente da pior para a melhor.

As estratégias de seleção de soluções mais adequadas a cada um dos agentes foram de modo geral, exceto para casos específicos, as seguintes:

SUBG : seleção de soluções duais com distribuição uniforme de probabilidade e seleção, sempre, da melhor solução primal;

BR : seleção de soluções primais com distribuição linear de probabilidade, crescente da pior para a melhor;

InitMP : seleção de soluções duais com distribuição linear de probabilidade, crescente da pior para a melhor;

GRP : seleção de soluções duais com distribuição linear de probabilidade, crescente da pior para a melhor;

LS : seleção de soluções primais com distribuição linear de probabilidade, crescente da melhor para a pior;

PERT : seleção de soluções primais com distribuição uniforme de probabilidade;

CONS : seleção de soluções duais com distribuição linear de probabilidade, crescente da pior para a melhor, e seleção de soluções primais com distribuição linear de probabilidade, crescente da pior para a melhor.

O esquema de construção de cortes de Bellmore e Ratliff [BR71] permite que, a partir de uma mesma solução primal x não redundante, sejam gerados até $\delta = |q_1| * |q_2| * \ldots * |q_k|$ cortes diferentes, onde k é o número de variáveis na solução com valor $x_j = 1$ e $|q_j|$ é o número de restrições saturadas por $|x_j|$. Assim,

a estratégia adotada no agente BR foi a construção de apenas um subconjunto desses cortes, de cardinalidade equivalente a um percentual de δ.

6.3 Instâncias de Testes

A aplicabilidade de *A-Teams* na resolução do SCP foi testada em seis classes de instâncias. Estas foram escolhidas de forma a abranger um vasto espectro de problemas, desde os de resolução relativamente fácil até aqueles reconhecidamente difíceis de serem resolvidos. As seis classes são compostas de instâncias:

1. geradas conforme proposto por Lucena [Luc94]. Neste esquema de geração, dado um conjunto I de pontos no plano, escolhem-se aleatoriamente n subconjuntos I_j de I com cardinalidades proporcionais à densidade desejada para a instância. A cada subconjunto I_j é associado um custo c_j, o qual corresponde ao peso da árvore geradora de custo mínimo calculada em cima dos pontos de I_j, considerando-se distâncias euclidianas;

2. selecionadas entre as utilizadas por Balas e Ho [BH80] nos testes computacionais de seu algoritmo. Os dois autores utilizaram 6 conjuntos de instâncias nesses testes. Assim, foram considerados os conjuntos 4, 5 e 6 para a escolha de um grupo de seis instâncias (4.1, 4.9, 5.2, 5.5, 6.4 e 6.5);

3. geradas a partir do cálculo de *1-Width* de matrizes de incidência de sistemas de triplas de Steiner. Estes problemas foram propostos por Fulkerson, Nemhauser e Trotter [FNT74];

4. geradas conforme esquema descrito por Balas e Ho [BH80]. Os problemas são gerados aleatoriamente de acordo com a densidade desejada, sujeitos a restrição de que cada subconjunto contenha no mínimo um elemento e cada elemento pertença a pelo menos dois conjuntos. Os custos associados aos subconjuntos são escolhidos aleatoriamente no intervalo $[1, \ldots, 100]$. Este tipo de problema foi largamente utilizado por vários autores em testes de algoritmos, como, por exemplo, Salkin e Koncal [SK73], Balas e Ho [BH80], Beasley [Bea87] e Fisher e Kedia [FK90].

5. como descritas no item anterior, porém fixando-se custos unitários aos subconjuntos; e

6. como descritas no item 4, porém associando-se a cada elemento um ponto no plano e calculando-se os custos dos subconjuntos conforme o esquema descrito no item 1.

6.4 Resultados Computacionais

As tabelas mostradas a seguir sumarizam os resultados dos testes realizados, com o *A-Team* mostrado na figura 6.1, sobre instâncias das classes descritas anteriormente. Os testes foram realizados em uma estação de trabalho *SPARC Server 1000* com oito processadores, 192Mb de memória principal e 311Mb de memória virtual, rodando o sistema operacional SunOS 5.5. Em todas as tabelas as colunas $d(\%)$ e Z_{lp} indicam respectivamente a densidade da matriz e o valor ótimo da relaxação contínua correspondentes à instância associada a linha da tabela. Z_{lb} e Z_{ub} correspondem aos valores obtidos para os limites inferior e superior, respectivamente, em cada instância. CPU, quando presente, indica o tempo de processamento até a obtenção destes limites. Na execução do *A-Team* foi fixado um limite de tempo (3600s na maioria dos testes realizados) como condição de parada. A coluna *cortes* indica o número de desigualdades acrescentadas ao problema original para a obtenção de Z_{lb}.

Em relação aos limites inferiores é importante ressaltar que eles foram gerados (em todos os testes realizados) a partir da resolução do dual da relaxação contínua do problema. Nesse processo foi utilizado o método de otimização por subgradientes, o qual, quase sempre, gera soluções infactíveis. Assim, faz-se necessária a projeção da solução gerada no espaço de soluções factíveis. Com isto há sempre uma perda no valor da função objetivo.

Os resultados de testes realizados sobre sete instâncias da classe 1, com o *A-Team* mostrado na figura 6.1, são apresentados na tabela 6.1. Para esta classe de instâncias também foram realizados testes com o algoritmo heurístico de Beasley [Bea90] e com o algoritmo exato de Fisher e Kedia [FK90]. Estes foram implementados nas linguagens C e FORTRAN-77, respectivamente. Na execução do algoritmo de Fisher e Kedia, assim como para o *A-Team*, também foi adotado um limite de tempo (3600s nos testes realizados) como condição de parada.

					A-Team		Beasley		F & K	
	$\|I\|$	$\|J\|$	d(%)	Z_{lp}	Z_{ub}	CPU	Z_{ub}	CPU	Z_{ub}	CPU
1.1	100	200	5.0	2368.7	2723.0	317	3081.0	2	2739.0	3642
1.2	200	400	2.0	5364.6	6040.0	83	7020.0	6	6077.0	3601
1.3	200	500	2.0	5167.5	5893.0	668	6790.0	9	5922.0	3604
1.4	200	1000	3.0	375.9	461.5	372	568.3	18	476.0	3634
1.5	300	1000	3.0	556.7	788.7	211	1009.2	29	804.9	3642
1.6	400	1000	3.0	657.0	1011.1	2	1280.8	40	993.8	3749
1.7	500	1000	3.0	769.5	1222.3	2398	1559.2	60	1249.8	3652

Tabela 6.1: Resultados obtidos para instâncias da classe 1, com a configuração do *A-Team* mostrada na figura 6.1

Uma análise dos resultados apresentados permite verificar a dificuldade de obter boas soluções para a classe de instâncias testadas. Observa-se que mesmo uma heurística sofisticada como a de Beasley não consegue soluções de boa qualidade com um tempo reduzido de CPU. O algoritmo enumerativo de Fisher e Kedia também conta com boas heurísticas, entretanto a qualidade das soluções que obtém evolui somente a medida que fixações são efetuadas, o que decorre em um dispendioso tempo de processamento.

O *A-Team* aqui proposto obteve a melhor solução em seis das sete instâncias, sendo que as suas melhores soluções são obtidas na maioria das vezes dentro dos primeiros dez minutos de processamento. Estes resultados indicam que a abordagem por Times Assíncronos gera resultados que são pelo menos comparáveis com alguns dos melhores algoritmos para o *SCP*.

A tabela 6.2 apresenta os limites inferiores e superiores, para duas das sete instâncias dessa classe, obtidos em três diferentes execuções do *A-Team*. Nestas o limitante de tempo foi fixado em 10800s. Estes resultados ilustram bem a dificuldade de se ajustar certos parâmetros do *A-Team* para diferentes instâncias. Cada uma das três execuções, para a instância 1.1, foi realizada com diferentes tamanhos de memórias e número máximo de cortes possíveis de serem adicionados ao problema. Nos três testes o valor de Z_{ub} foi igual ao da solução ótima, e o de Z_{lb} foi, em média, 10% superior ao de Z_{lp}. Nos outros testes, problema 1.3, foram utilizadas as mesmas configurações para os parâmetros citados, ajustados proporcionalmente às dimensões da instância. Os resultados assim obtidos mostraram-se totalmente irregulares, conforme observado no gráfico da figura 6.2, o qual mostra a evolução dos limites superiores, nas três execuções, nos três primeiros minutos de processamento.

					1ª Execução		2ª Execução		3ª Execução	
	$\|I\|$	$\|J\|$	d(%)	Z_{lp}	Z_{lb}	Z_{ub}	Z_{lb}	Z_{ub}	Z_{lb}	Z_{ub}
1.1	100	200	5.0	2368.7	2533.4	2723.0	2563.9	2723.0	2596.7	2723.0
1.3	200	500	2.0	5167.5	5166.8	5814.0	5080.9	5923.0	5079.2	5889.0

Tabela 6.2: Resultados de três execuções do *A-Team* para duas instâncias da classe 1

As instâncias da classe 2, consideradas relativamente fáceis de serem resolvidas, foram testadas fixando-se o tempo de execução do *A-Team* em apenas 1800s. Os resultados obtidos estão na tabela 6.3. Em apenas uma das instâncias (2.2) o limite Z_{ub} não foi igual ao valor da solução ótima. Em outra (2.3) o limite obtido foi melhor do que o valor da melhor solução conhecida. Em todas as instâncias (exceto 2.1 e 2.4) o valor de Z_{lb} ultrapassou o valor de Z_{lp}. O gráfico da figura 6.3 mostra a evolução dos limites obtidos ao longo do tempo de execução, para a

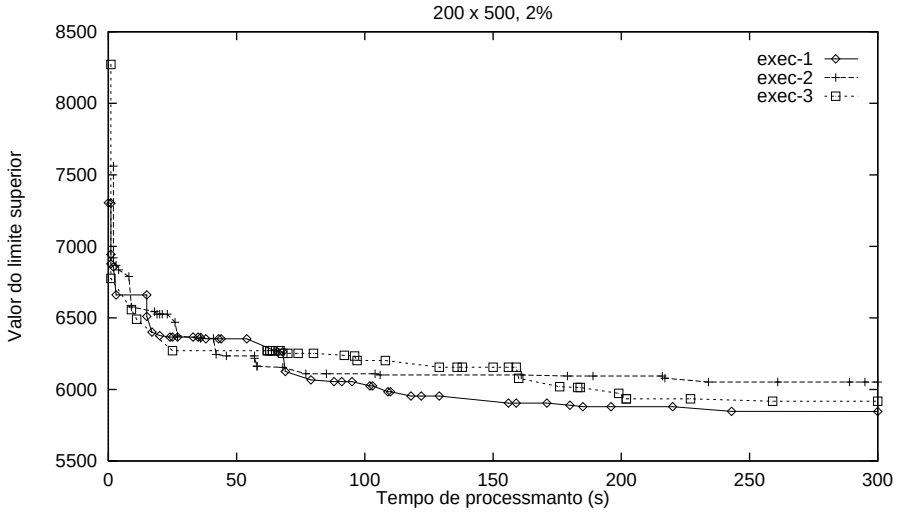

Figura 6.2: Evolução do limite superior nos três testes para a instância 1.3

instância 2.1. É importante ressaltar que o número de cortes efetivamente acrescentados ao problema original, antes da geração de soluções duais, foi bastante reduzido, dentro do limite máximo de 50.

	$\|I\|$	$\|J\|$	d(%)	Z_{lp}	Z^*	Z_{lb}	$cortes$	Z_{ub}
2.1 (4.1)	200	1000	2.0	429.0	429.0	429.0	06	429.0
2.2 (4.9)	200	1000	2.0	636.6	641.0	641.0	22	644.0
2.3 (5.2)	200	2000	2.0	299.3	307.0[a]	299.8	13	302.0
2.4 (5.5)	200	2000	2.0	211.0	211.0	211.0	17	211.0
2.5 (6.4)	200	1000	5.0	128.1	131.0	129.0	32	131.0
2.6 (6.5)	200	1000	5.0	152.5	161.0	153.4	27	161.0

[a] Melhor solução conhecida

Tabela 6.3: Resultados obtidos para seis instâncias da classe 2

A tabela 6.4 mostra os resultados obtidos para um conjunto de oito instâncias da classe número 4. Devido a similaridade destas com as da classe 2, foram utilizadas as mesmas configurações de tamanhos de memórias usadas para os testes com esta classe. Os limites superiores obtidos (Z_{ub}) igualam, em sete das instâncias, ao valor da solução ótima (Z^*). Na única exceção (instância 4.5) a diferença entre os dois valores foi inferior a 1%. As diferenças entre os limites inferiores (Z_{lb}) e os valores das soluções ótimas para a relaxação linear dos problemas (Z_{lp}) foram inferiores a 2%. Aqui também, o desempenho com relação aos limites superiores pode ser considerado muito bom. Os limites inferiores, aparentemente, poderiam ser melhorados alterando-se os valores de algums parâmetros, tais como, o número de cortes adicionados ao problema original. A justificativa para tal dedução é que a alteração de alguns dos parâmetros gerou bons resultados nos

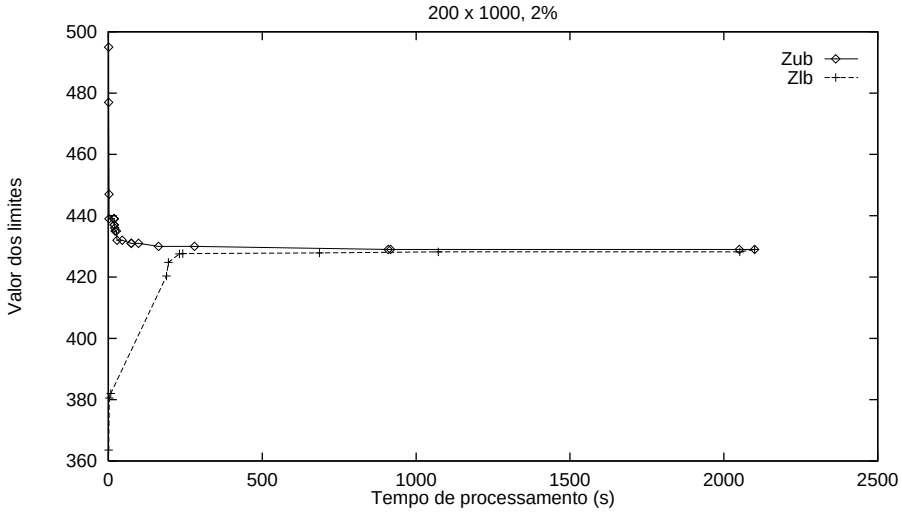

Figura 6.3: Evolução dos limites inferior e superior, para a instância 2.1, durante a execução do *A-Team* mostrado na figura 6.1

testes realizados com as instâncias da classe 2, que são bastante similares às da classe 4.

	$\|I\|$	$\|J\|$	d(%)	Z_{lp}	Z^*	Z_{lb}	Z_{ub}
4.1	200	500	2.0	927.6	932.6	924.1	932.6
4.2	200	500	2.0	907.3	909.5	904.8	909.5
4.3	200	1000	2.0	497.6	497.8	497.8	497.8
4.4	200	1000	2.0	433.3	433.5	432.9	433.5
4.5	200	500	5.0	345.8	373.4	341.5	375.6
4.6	200	500	5.0	235.2	242.8	230.6	242.8
4.7	200	1000	5.0	132.1	135.3	129.7	135.3
4.8	200	1000	5.0	127.4	131.1	124.6	131.1

Tabela 6.4: Resultados obtidos para oito instâncias da classe 4

As tabelas 6.5 e 6.6 mostram os resultados obtidos para o mesmo conjunto de oito instâncias citadas anteriormente, porém, com custos unitários associados aos subconjuntos (tabela 6.5) e recalculados de acordo com o esquema proposto por Lucena [Luc94] (tabela 6.6). Observe que os resultados descritos nas tabelas 6.5 e 6.6 referem-se a instâncias das classes 5 e 6, respectivamente. Como as dimensões das instâncias são as mesmas das da classe 4, estes testes foram realizados com as mesmas configurações do *A-Team*. Os resultados apresentados para essas duas classes, confirmam a dificuldade de ajuste dos parâmetros.

Os resultados obtidos para a classe 3, de resolução reconhecidamente difícil, estão mostrados na tabela 6.4. As diversas variações testadas na arquitetura do *A-Team* não conseguiram encontrar, nem mesmo para a instância 3.1 de menor

	$\|I\|$	$\|J\|$	d(%)	Z_{lp}	Z_{lb}	Z_{ub}
5.1	200	500	2.0	39.5	38.6	45.0
5.2	200	500	2.0	38.2	37.1	43.0
5.3	200	1000	2.0	33.3	32.6	40.0
5.4	200	1000	2.0	33.0	32.4	39.0
5.5	200	500	5.0	16.5	15.8	23.0
5.6	200	500	5.0	16.7	16.1	24.0
5.7	200	1000	5.0	14.6	14.0	21.0
5.8	200	1000	5.0	14.0	13.4	20.0

Tabela 6.5: Resultados obtidos para oito instâncias da classe 5

	$\|I\|$	$\|J\|$	d(%)	Z_{lp}	Z_{lb}	Z_{ub}
6.1	200	500	2.0	5013.5	4950.7	5507.0
6.2	200	500	2.0	4905.2	4853.2	5175.0
6.3	200	1000	2.0	4202.0	4168.7	4548.0
6.4	200	1000	2.0	4216.2	4181.0	4580.0
6.5	200	500	5.0	3762.5	3628.7	5139.0
6.6	200	500	5.0	3827.8	3702.7	5078.0
6.7	200	1000	5.0	3481.9	3362.8	4904.0
6.8	200	1000	5.0	3499.7	3433.1	4880.0

Tabela 6.6: Resultados obtidos para oito instâncias da classe 6

porte, limites inferiores melhores do que o valor de Z_{lp}, embora os limites superiores encontrados para todas as instâncias sejam iguais (3.1 e 3.3) ou pouco superiores (3.2, 3.4 e 3.5) aos das melhores soluções conhecidas (veja [FNT74] e [FR95]). É importante ressaltar aqui que, para todas as instâncias, o limite inferior Z_{lb} foi obtido pelo agente GRD. Portanto, para essa classe de instâncias, as soluções obtidas pela variação proposta na seção 3.2.1 para a heurística gulosa dual ali descrita, mostraram-se dominantes em relação às obtidas pelo método dos subgradientes.

	$\|I\|$	$\|J\|$	d(%)	Z_{lp}	Z_{lb}	Z_{ub}
3.1	117	27	11.1	9.0	9.0	19.0[a]
3.2	330	45	6.7	15.0	14.9	31.0
3.3	1080	81	3.7	27.0	27.0	61.0[a]
3.4	3015	135	2.2	45.0	44.9	105.0
3.5	9801	243	1.2	81.0	80.9	205.0

[a] Solução ótima

Tabela 6.7: Resultados obtidos para quatro instâncias da classe 3

Capítulo 7

Conclusões

7.1 Conclusões

Esta dissertação apresenta um estudo sobre algumas características estruturais do problema de Recobrimento de um Conjunto, os principais métodos computacionais para sua resolução e propõe a aplicação de uma nova abordagem heurística, multi-algorítmica, ao mesmo.

A abordagem utilizada, Times Assíncronos, no desenvolvimento de uma arquitetura aplicada ao SCP, mostrou-se extremamente sensível a variações em diversos parâmetros, tais como, tamanhos de memórias, tempo de execução e número de agentes utilizados. Os exaustivos testes realizados em diversas classes de instâncias sugerem que um comportamento similar pode ocorrer para outros problemas. A causa aparente de tal comportamento da metodologia é o grande número de diferentes classes de instâncias do problema. Este fator torna difícil a busca de uma configuração única para diferentes classes.

A despeito dos resultados obtidos, a abordagem primal-dual com a utilização de cortes certamente é promissora. Algumas observações que levam a tal conclusão são:

1. os métodos heurísticos utilizados para a geração de soluções primais mostraram-se robustos, ou seja, eles conseguiram gerar as soluções ótimas, ou bem próximas disto, para a quase totalidade das instâncias testadas, independentemente a que classe pertenciam;

2. a obtenção de soluções primais, por heurísticas gulosas que utilizam variáveis duais no processo de construção das soluções, mostrou-se sensível à melhora das soluções duais;

3. o uso da relaxação Lagrangeana e do método de otimização por subgradientes consegue produzir boas soluções duais; e

4. o uso de cortes, efetivamente, auxilia as heurísticas duais na obtenção de soluções de melhor valor objetivo.

7.2 Extensões e Trabalhos Futuros

Assumiu-se, no desenvolvimento deste trabalho, que o enfoque principal seria dado aos métodos computacionais para a resolução do SCP. Assim, os seguintes comentários são concernentes a respeito do conteúdo desta dissertação:

- **Capítulo I :**

  O estudo mais aprofundado de aplicações do problema não foi considerado como relevante ao presente trabalho. Contudo, sem dúvida alguma, a extensa literatura a respeito de aplicações do SCP a diversos problemas práticos permitiria que este estudo fosse realizado. Assim, uma extensão a este trabalho é o mapeamento desses diversos problemas e suas modelagens como SCP.

- **Capítulo II :**

  Na seção 2.2.3 esboçou-se a importância do SCP na resolução de problemas advindos da Teoria dos Grafos. Além dos problemas ali citados existem vários outros cujas resoluções podem ser feitas a partir do SCP. A enumeração desses problemas, suas modelagens como SCP e análises a respeito da eficiência dessas modelagens é outra possível extensão ao presente trabalho.

  Uma pequena revisão bibliográfica a respeito da estrutura facial do SCP foi feita na seção 2.5. As referências ali contidas são apenas o ponto de partida para um estudo mais apronfundado nessa área.

  Dois dos principais problemas relacionados ao SCP, o SPP e o SP, foram mostrados na seção 2.2.1. Limitou-se ali a descrição destes dois problemas e suas modelagens como problemas de programação linear inteira. O estudo da estrutura facial desses problemas, principalmente do SP, complementaria o mesmo estudo a respeito do SCP.

- **Capítulo III :**

  Existem na literatura várias propostas de aplicação de meta-heurísticas, tais como "*Tabu Search*", "*Simulated Annealing*" e "*GRASP*", ao SCP ([FR95, BC94, Sen93, JB93]). Considerando-se que Times Assíncronos também são meta-heurísticas, seria de interesse um estudo comparativo destes com as outras abordagens.

  Observou-se nos testes realizados que o desempenho do método de otimização por subgradientes pode ser irregular, dependendo da classe de instâncias tratada e do número de cortes adicionados ao problema. Um estudo mais aprofundado desses fatores certamente auxiliaria no processo de convergência dos limites inferiores.

- **Capítulo IV :**

  Explorou-se no trabalho a estrutura de duas classes de algoritmos: enumerativos e baseados em planos de cortes. Existem outras classes de algoritmos, menos utilizados no caso do SCP, que também poderiam ser analisadas, como por exemplo algoritmos baseados em classificação e aglutinação de subconjuntos ("*Group Theoretic Approach*").

  Assumiu-se nesse capítulo como bem conhecidas as estruturas dos algoritmos de busca e baseados em planos de cortes. Contudo, uma discussão mais detalhada dos mesmos seria totalmente complementar ao conteúdo do capítulo.

  Os algoritmos descritos foram propostos ao longo das ultimas 3 décadas. Nos testes computacionais dos mesmos foi utilizada uma grande diversidade de instâncias e equipamentos. Assim, torna-se difícil extrair alguma informação realmente útil através da comparação dos resultados apresentados. Resultados conclusivos só poderiam ser obtidos com a implementação do algoritmos e realização de testes para um mesmo conjunto de instâncias.

- **Capítulo V :**

  Conforme observado na seção 5.3 os modelos de *A-Teams* ali propostos para o SCP não são únicos. Existem diversas alternativas viáveis que podem ser utilizadas. Assim, uma extensão é a exploração de variações dos modelos propostos e proposição de novas alternativas. Poderia-se, por exemplo, propor modelos específicos para classes particulares de instâncias.

Analisando-se os resultados computacionais obtidos com o *A-Team* proposto, conclui-se que estes poderiam ser melhorados introduzindo-se algumas modificações na estrutura do modelo. Dentre essas modificações identificou-se as seguintes mudanças e extensões no modelo, e na sua implementação, como mais as mais importantes:

1. Melhorar os critérios utilizados para a exclusão de cortes armazenados na memória. Uma idéia é a fixação de um limite mínimo (um obtido, por exemplo, com o método dos subgradientes sem o uso de cortes) e a associação de penalidades aos cortes. Assim os cortes que fossem utilizados, e conseguissem contribuir para a ultrapassagem desse limite mínimo, teriam menor probabilidade de ser excluídos. O intuito dessa alteração é que os cortes mais "fracos" sejam excluídos e os mais "fortes" sejam mantidos na memória.

2. Utilizar os cortes gerados pelo agente BR (Bellmore e Ratliff) no lado primal do *A-Team*, ou seja, acrescentar os cortes ao problema antes da utilização

dos agentes geradores/modificadores de soluções primais. Como esses cortes eliminam a melhor solução conhecida até o momento da geração do corte, então com essa alteração diminui-se o espaço de busca de uma nova solução.

3. Testar a aplicação computacional de algumas classes de facetas já descritas para o politopo do SCP, ou seja, o seu uso como planos de cortes.

4. Dividir a memória de soluções primais, criando uma memória de soluções parciais. Utilizar diferentes critérios de análise de consenso entre soluções primais na construção de soluções parciais.

5. Estender o *A-Team* para utilizar a modelagem conjunta SCP-SPP proposta por Fisher e Kedia.

Bibliografia

[Bak81] E. K. Baker. Efficient Heuristic Algorithms for the Weighted Set Covering. *Computer and Operations Research*, 8(4):303–310, 1981.

[Bal80] E. Balas. Cutting Planes from Conditional Bounds: A New Approach to Set Covering. *Mathematical Programming*, 12:19–36, abril de 1980.

[BC94] J. E. Beasley e P. C. Chu. A Genetic Algorithm for the Set Covering Problem. Working paper, The Management School. Imperial College, London. SW7 2AZ. England., julho de 1994.

[Bea87] J. E. Beasley. An Algorithm for Set Covering Problem. *European Journal of Operational Research*, 31(1):85–93, julho de 1987.

[Bea90] J. E. Beasley. A Lagrangian Heuristic for Set-Covering Problems. *Naval Research Logistics*, 37(1):151–164, fevereiro de 1990.

[BH80] E. Balas e A. Ho. Set Covering Algorithms using Cutting Planes, Heuristics, and Subgradient Optimization: A Computational Study. *Mathematical Programming Study*, 12:37–60, abril de 1980.

[BJ92] J. E. Beasley e K. Jornsten. Enhancing an Algorithm for Set Covering Problems. *European Journal of Operational Research*, 58(2):293–300, abril de 1992.

[BN89a] E. Balas e S. M. Ng. On the Set Covering Polytope: I. All the Facets with Coefficients in {0,1,2}. *Mathematical Programming*, 43(1):57–69, janeiro de 1989.

[BN89b] E. Balas e S. M. Ng. On the Set Covering Polytope: II. Lifting the Facets with Coefficients in {0,1,2}. *Mathematical Programming*, 45(1):1–20, agosto de 1989.

[BP76] E. Balas e M. W. Padberg. Set partitioning: A survey. *SIAM Review*, 18(5):710–760, outubro de 1976.

[BR71] M. Bellmore e H. D. Ratliff. Set Covering and Involutory Bases. *Management Science*, 18(3):194–206, novembro de 1971.

[Car87] E. Cardozo. *DPSK: A Kernel fo Distributed Problem Solving*. PhD thesis, Electrical and Computer Engineering Department, Carnegie Mellon University, Pittsburgh, PA, 1987.

[Cav95] V. F. Cavalcante. Times Assíncronos para o *Job Shop Scheduling*: heurísticas de construção. Master's thesis, Departamento de Ciência da Computação, Universidade Estadual de Campinas, Campinas, SP, 1995.

[CFM75] P. M. Camerini, L. Fratta, e F. Maffioli. On Improving Relaxation Methods by Modified Gradient Techniques. *Mathematical Programming Study*, 3:26–34, 1975.

[Chv79] V. Chvátal. A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research*, 4(3):233–235, agosto de 1979.

[CK75] N. Christofides e S. Korman. A Computacional Survey of Methods for Set Covering Problem. *Management Science*, 21(5):591–599, janeiro de 1975.

[CS89] G. Cornuéjols e A. Sassano. On the 0,1 Facets of the Set Covering Polytope. *Mathematical Programming*, 43(1):45–55, janeiro de 1989.

[CS92] E. Cardozo e S. Sichman. *DPSK+P User's Manual - C++ Interface, version 1.0*. FEE/UNICAMP, outubro de 1992.

[Edm62] J. Edmonds. Covers and Packing in a Family of Sets. *Bulletin of American Mathematical Society*, 68(5):494–499, setembro de 1962.

[Edm65] J. Edmonds. Path, Trees, and Flowers. *Canadian Journal of Mathematics*, 17(3):449–467, 1965.

[EDM92] E. El-Darzi e G. Mitra. Solution of Set-Covering and Set-Partitioning Problems using Assignment Relaxations. *Journal of Operational Research Society*, 43(5):483–493, 1992.

[Erl78] D. Erlenkotter. A Dual-Based Procedure for Uncapacitated Facility Location. *Operations Research*, 26(6):992–1019, novembro de 1978.

[Etc77] J. Etcheberry. The Set Covering Problem: A New Implicit Enumeration Algorithm. *Operations Research*, 25(5):760–772, setembro de 1977.

[FH80] M. L. Fisher e D. S. Hochbaum. Database Location in Computer Networks. *Journal of the Association for Computing Machinery*, 27(4):718–735, outubro de 1980.

[Fis80] M. L. Fisher. Worst-Case Analysis of Heuristic Algorithms. *Management Science*, 26(1):1–17, janeiro de 1980.

[Fis81] M. L. Fisher. The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science*, 27(1):1–18, janeiro de 1981.

[Fis85] M. L. Fisher. An Applications Oriented Guide to Lagrangian Relaxation. *Interfaces*, 15(2):10–21, março de 1985.

[FK90] M. L. Fisher e P. Kedia. Optimal Solution of Set Covering/Partitioning Problems using Dual Heuristics. *Management Science*, 36(6):674–688, junho de 1990.

[FNT74] D. R. Fulkerson, G. L. Nemhauser, e L. E. Trotter. Two Computationally Difficult Set Covering Problems that Arise in Computing the 1-Width of Incidence Matrices of Steiner Triple Systems. *Mathematical Programming Study*, 2:72–81, 1974.

[FR61] D. R. Fulkerson e H. J. Ryser. Widths and Heights of (0,1)-Matrices. *Canadian Journal of Mathematics*, 13(2):239–255, 1961.

[FR89] T. A. Feo e M. G. C. Resende. A Probabilistic Heuristic for a Computationally Difficult Set Covering Problem. *Operations Research Letters*, 8:67–71, abril de 1989.

[FR95] T. A. Feo e M. G. C. Resende. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, ?:1–27, 1995.

[Ful71] D. R. Fulkerson. Blocking and Anti-Blocking Pairs of Polyhedra. *Mathematical Programming*, 1:168–194, 1971.

[FW82] M. L. Fisher e L. A. Wolsey. On the Greedy Heuristic for Continuous Covering and Packing Problems. *SIAM Journal on Algebraic and Discrete Methods*, 3(4):584–591, dezembro de 1982.

[Geo74] A. M. Geoffrion. Lagrangean Relaxation for Integer Programming. *Mathematical Programming Study*, 2:82–114, 1974.

[GJ79] M. R. Garey e D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York, 1979.

[Glo71] F. Glover. A Note on Extreme-Point Solutions and a Paper by Lemke, Salkin, and Spielberg. *Operations Research*, 19(1):1023–1025, julho de 1971.

[Glo89] F. Glover. Tabu Search - Part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[Glo90] F. Glover. Tabu Search - Part II. *ORSA Journal on Computing*, 2(1):4–32, 1990.

[GN69] R. S. Garfinkel e G. L. Nemhauser. The Set Partitioning Problem: Set Covering with Equality Constraints. *Operations Research*, 17(5):848–856, setembro de 1969.

[GN72] R. S. Garfinkel e G. L Nemhauser. *Integer Programming*. John Wiley & Sons, New York, 1972.

[Gof77] J. L. Goffin. On Convergence Rates of Subgradient Optimization Methods. *Mathematical Programming*, 13(2):329–347, dezembro de 1977.

[Gom63] R. Gomory. An Algorithm for Integer Solutions to Linear Programs. In Graves e Wolfe, editores, *Recent Advances in Mathematical Programming*. McGraw-Hill, 1963.

[GS79] M. Guinard e K. Spielberg. A Direct Dual Method for the Mixed Plant Locations Problem with Some Side Constraints. *Mathematical Programming*, 17(2):198–228, setembro de 1979.

[Guh73] D. Guha. The Set Covering with Equality Constraints. *Operations Research*, 21(1):348–351, janeiro de 1973.

[HK70] M. Held e R. M. Karp. The Traveling-Salesman Problem and Minimum Spanning Trees. *Operations Research*, 18(6):1138–1162, novembro de 1970.

[HK71] M. Held e R. M. Karp. The Traveling-Salesman Problem and Minimum Spanning Trees : Part II. *Mathematical Programming*, 1(1):6–25, outubro de 1971.

[HNR66] R. W. House, L. D. Nelson, e T. Rado. Computer Studies of a Certain Classes of Linear Integer Problems. In A. Lavi e T. P. Vogel, editores, *Recent Advances in Optimization Techniques*, pp. 241–280. John Wiley & Sons, 1966.

[Ho82] A. C. Ho. Worst Case Analysis of a Class of Set Covering Heuristics. *Mathematical Programming*, 23(2):170–180, junho de 1982.

[Hoc82] D. S. Hochbaum. Approximation Algorithms for the Set Covering and Vertex Cover Problems. *SIAM Journal on Computing*, 11(3):555–556, agosto de 1982.

[HWC74] M. Held, P. Wolfe, e H. P. Crowder. Validation of Subgradient Optimization. *Mathematical Programming*, 6(1):62–88, fevereiro de 1974.

[JB93] L. W. Jacobs e M. J. Brusco. A Simulated Annealing-Based Heuristic for the Set-Covering Problem. Technical report, Operations Management and Information Systems Department. Northern Illinois University, Dekalb. IL 60115. USA, março de 1993.

[Joh74] D. S. Johnson. Approximation Algorithms for Combinatorial Problems. *Journal of Computer and System Sciences*, 9(3):256–278, dezembro de 1974.

[Kar72] R. M. Karp. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*, pp. 85–104, New York, 1972. Plenum Press.

[KS90] R. K. Kwatera e B. Simeone. Clustering Heuristics for Set Covering. Technical report, Dipartimento de Statistica, Probabilità e Statistiche Applicate, Universitá "La Sapienza", Roma - Italia, 1990.

[Lau89] M. Laurent. A Generalization of Anti-Webs to Independence Systems and their Canonical Facets. *Mathematical Programming*, 45:97–108, 1989.

[Law66] E. L. Lawler. Covering Problems : Duality Relations and a New Method of Solution. *SIAM Journal Applied Mathematics*, 14(5), setembro de 1966.

[Lov75] L. Lovász. On the Ratio of Optimal and Fractional Covers. *Discrete Mathematics*, 13(4):383–390, dezembro de 1975.

[LSS71] C. Lemke, H. Salkin, e K. Spielberg. Set Covering by Single Branch Enumeration. *Operations Research*, 19(4):998–1022, julho de 1971.

[Luc94] A. Lucena. Comunicação Pessoal, 1994.

[Mar74] R. E. Marsten. An Algorithm for Large Set Partitioning Problems. *Management Science*, 20(5):774–787, janeiro de 1974.

[NR59] R. Z. Norman e M. O. Rabin. An Algorithm for a Minimum Cover of a Graph. *Proceeding of the American Mathematical Society*, 10(2):315–319, 1959.

[NS89] P. Nobili e A. Sassano. Facets and Lifting Procedures for the Set Covering Polytope. *Mathematical Programming*, 45(1):111–137, agosto de 1989.

[NS92] P. Nobili e A. Sassano. A Separation Routine for the Set Covering Problem. In *Proc. II IPCO - Integer Programming and Combinatorial Optimization*, Carnegie Mellon University, Pittsburgh, PA, maio de 1992.

[NT74] G. Nemhauser e L. Trotter. Properties of Vertex Packing and Independence System Polyhedra. *Mathematical Programming*, 6:48–61, 1974.

[NW88] G. L. Nemhauser e L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1988.

[Pad79] M. W. Padberg. Covering, Packing and Knapsack Problems. *Annals of Discrete Mathematics*, 4:265–287, 1979.

[Pei95] H. P. Peixoto. Metodologia de Especificação de Times Assíncronos para Problemas de Otimização Combinatória. Master's thesis, Departamento de Ciência da Computação, Universidade Estadual de Campinas, Campinas, SP, 1995.

[Pie68] J. F. Pierce. Application of Combinatorial Programming to a Class of All-Zero-One Integer Programming Problems. *Management Science*, 15(3):191–209, novembro de 1968.

[PL73] J. F. Pierce e J. Lasky. Improved Combinatorial Programming Algorithms for a Class of All Zero-One Integer Programming Problems. *Management Science*, 19(5):528–543, janeiro de 1973.

[Rot69] R. Roth. Computer Solutions to Minimum-Cover Problems. *Operations Research*, 17(3):455–465, maio de 1969.

[Sas89] A. Sassano. On the Facial Structure of the Set Covering Polytope. *Mathematical Programming*, 44(2):181–202, junho de 1989.

[Sen93] S. Sen. Minimal Cost Set Covering using Probabilistic Methods. In *Proc. 1993 ACM/SIGAPP Symposium on Applied Computing*, pp. 157–164, 1993.

[Sha79] J. F. Shapiro. A Survey of Lagrangean Techniques for Discrete Optimization. *Annals of Discrete Mathematics*, 5:113–138, 1979.

[SK73] H. M. Salkin e R. D. Koncal. Set Covering by an All Integer Algorithm : Computacional Experience. *Journal of the Association for Computing Machinery*, 20(3):189–193, abril de 1973.

[SM89] H. M. Salkin e K. Mathur. *Foundations of Integer Programming*. North-Holland, 1989.

[Sou93] P. S. Souza. *Asynchronous Organizations for Multi-Algorithm Problems*. PhD thesis, Department of Electrical and Computer Engineering of Carnegie Mellon University, Pittsburgh, PA, abril de 1993.

[ST93] P. S. Souza e S. N. Talukdar. Asynchronous Organizations for Multi-Algorithm Problems. In *ACM Symposium on Applied Computing*, Indianapolis, IN, fevereiro de 1993.

[Tah75] H. A. Taha. *Integer Programming: Theory, Applications and Computations*. Academic Press, 1975.

[Tal93] S. N. Talukdar. Asynchronous Teams. In *Fourth Symposium on Expert Systems Application to Power Systems*, Melbourne, Australia, novembro de 1993.

[TS90] S. N. Talukdar e P. S. Souza. Asyncrhonous Teams. In *Second SIAM Conference on Linear ALgebra : Signals, Systems and Control*, San Francisco, CA, novembro de 1990.

[TS92] S. N. Talukdar e P. S. Souza. Scale Efficienty Organizations. In *IEEE Int. Conference on Systems, Man and Cybernetics*, Chicago, IL, outubro de 1992.

[TS93] S. N. Talukdar e P. S. Souza. Objects Organizations and Super-Agents. In *Engineering Design : The Creation of Products and Processes*. MacGraw Hill, 1993.

[VW84] F. J. Vasko e G. R. Wilson. An Efficient Heuristic for Large Set Covering Problems. *Naval Research Logistics Quaterly*, 31(2):163–171, maio de 1984.

[VW88] F. J. Vasko e F. E. Wolf. Solving Large Set Covering Problems on a Personal Computer. *Computer and Operations Research*, 15(2):115–121, 1988.

[Wol80] L. A. Wolsey. Heuristic Analysis, Linear Programming and Branch and Bound. *Mathematical Programming Study*, 13:121–134, agosto de 1980.

[Won84] R. T. Wong. A Dual Ascent Approach for Steiner Tree Problems on a Directed Graph. *Mathematical Programming*, 28(3):271–287, abril de 1984.